ABSTRACT

Title of dissertation: MEASUREMENT-BASED OPTIMAL ROUTING STRATEGIES ON OVERLAY ARCHITECTURES

Tuna Güven, Doctor of Philosophy, 2006

Dissertation directed by: Professor Mark A. Shayman
Department of Electrical and Computer Engineering

Professor Richard J. La
Department of Electrical and Computer Engineering

As a natural outcome of dramatic growth of the Internet and proliferation of demanding services with higher data rates, traffic engineering has proved to be vital in avoiding congestion and maximizing the network resource utilization. A key component of traffic engineering is traffic mapping, *i.e.,* splitting the traffic flows along multiple paths. In this thesis, we seek optimal, yet practical, multipath routing algorithms that can minimize the network congestion by exploiting the locally collected measurement data.

We first develop a distributed measurement-based routing algorithm to load balance intradomain traffic along multiple paths for multiple unicast sources. Multiple paths are established using overlay nodes. The algorithm is derived from simultaneous perturbation stochastic approximation (SPSA) and does not assume that the gradient of an analytical cost function is known. Instead, it relies on (potentially) noisy estimates from local measurements. We formulate the traffic mapping

problem in an optimization framework and show through an analytical model that the algorithm converges to the optimal solution almost surely under a decreasing step size policy (as with the standard SPSA model). Motivated by practical concerns, we next consider the constant step size case, for which we establish weak convergence.

In the second part of this thesis, we consider the problem of load balancing of multicast traffic sessions and generalize our unicast routing algorithm to route both types of traffic simultaneously. We consider three network models that reflect different sets of assumptions regarding multicast capabilities of the network. As in the unicast case, we prove the almost sure convergence of the algorithm to a corresponding optimal solution under each network model considered with decreasing step sizes and establish the weak convergence with a fixed step size. In addition, we investigate the benefits acquired from implementing additional multicast capabilities by studying the relative performance of the generalized algorithm under the three network models.

Throughout this thesis, we rely on an overlay architecture to establish multiple paths between a source and its destination(s) in an IP network. As the performance of the routing algorithms depends on the quality of paths provided by the overlay nodes, it is of interest to carefully locate a limited number of overlay nodes in the network. The final part of this thesis makes use of the discrete stochastic optimization methods and presents an optimal solution based on Stochastic Comparison (SC) algorithm to locate overlay nodes given a set of sources and their corresponding destination(s). Motivated by the impracticality of stochastic comparison algorithm in

an online setting due to its computational complexity, we provide a computationally efficient heuristic solution. We show through a detailed simulation study that the performance obtained by the heuristic solution is comparable to that of the optimal algorithm.

MEASUREMENT-BASED OPTIMAL ROUTING STRATEGIES ON OVERLAY ARCHITECTURES

by

Tuna Güven

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Commmittee:

Professor Mark A. Shayman, Chair/Advisor
Professor Richard J. La, Co-advisor
Professor Armand Makowski
Professor Bobby Bhattacharjee
Professor Samir Khuller

DEDICATION

To my parents and to Anna.

ACKNOWLEDGMENTS

First, I would like to thank my advisors Professor Mark Shayman and Professor Richard J. La. Their guidance, encouragement and patience were invaluable for me to write this thesis as well as to establish myself as a better researcher. It has been an honor for me to work with and learn from them.

I would like to thank Professor Bobby Bhattacharjee for sharing his time and opinion during our weekly group meetings. His suggestions was essential for me to have better publications. I also thank the members of my committee, Professors Armand Makowski and Samir Khuller for reading this thesis and for their insightful comments.

I would especially like to thank my love, Anna Pantelidou for making this journey so special. Her unconditional love, inexpressible encouragement and never-ending belief in me, have given me the power to climb over "this mountain". It is this inspiration she has given me that will allow me to climb many more "mountains" awaiting us.

I would also like to thank my parents Semra and Mustafa Guven, and my brother Tolga Guven. It was their endless love, care and encouragement that made me stood up and carry on in the most difficult times.

I have been very fortunate to meet and made many friends at Maryland. With their friendship and support I have had such a good time during all these years at

Maryland as well as so many memories to remember: Tolga Girici, Onur Kaya, Volkan Efe, Cemal Yilmaz, Onur Ergin and many others that I forgot to mention here.

TABLE OF CONTENTS

List of Tables vii

List of Figures viii

1 Introduction 1

2 Measurement Based Optimal Multipath Routing for Unicast Sessions 10
2.1 Model . . . 10
2.2 Stochastic Approximation . . . 12
2.3 Optimal Routing Using SPSA . . . 15
2.3.1 An Optimal Routing Algorithm - Decreasing Step Size . . . 15
2.3.2 The Optimal Routing Algorithm - Constant Step Size . . . 18
2.3.3 Measurement Process . . . 20
2.4 Implementation Issues . . . 21
2.4.1 Path Establishment . . . 22
2.4.2 Traffic Monitoring . . . 23
2.5 Experimental Setup and Simulation Results . . . 24
2.6 Conclusion . . . 34
2.7 Proof of Theorem 2.3.1 . . . 35
2.8 Proof of Theorem 2.3.2 . . . 42

3 **A Unified Framework for Multipath Routing for Unicast and Multicast Traffic** 43
3.1 Background & Set-up . . . 43
3.1.1 Digital Fountain codes . . . 44
3.2 Network models . . . 47
3.2.1 Network Model-I . . . 48
3.2.2 Network Model-II . . . 49
3.2.3 Network Model-III . . . 50
3.3 Optimization Framework and the Proposed Algorithm . . . 52
3.3.1 Proposed routing algorithm . . . 53
3.4 Convergence of the Proposed Algorithm . . . 55
3.4.1 Decreasing step size case . . . 56
3.4.2 Constant step size case . . . 59
3.5 Simulation Results . . . 61
3.5.1 First topology . . . 62
3.5.2 Second topology . . . 67
3.6 Conclusion . . . 73
3.7 Proof of Theorem 3.4.2 . . . 75
3.8 Proof of Theorem 3.4.4 . . . 80
3.9 Proof of Lemma 3.7.1 . . . 81

4 Obtaining Better Paths through Overlay Node Selection 87
4.1 Introduction 87
4.2 Model 88
4.3 Stochastic Comparison Algorithm 90
4.4 A suboptimal heuristic 93
4.5 Simulation Results 94
4.5.1 Unicast traffic sessions under first network topology 96
4.5.2 Unicast traffic sessions under second topology 98
4.5.3 Unicast traffic sessions with a non-uniform traffic matrix . . . 101
4.5.4 Multicast traffic sessions 102

5 Conclusion 105

Bibliography 108

LIST OF TABLES

2.1 The Cross Traffic Dynamics 27

4.1 Performance comparison of GH vs. SC. 98

4.2 Performance comparison of GH vs. SC. 100

4.3 Performance comparison of GH vs. SC. 102

4.4 Performance comparison of GH vs. SC. for multicast traffic 104

LIST OF FIGURES

2.1 Proposed overlay architecture for multipath routing. 22

2.2 Network Topology 1 26

2.3 Offered Load on Network Topology 1 with an offset of 50 ms 29

2.4 Total packet drops on Network Topology 1 with an offset of 50 ms . . 29

2.5 Offered Load on Network Topology 1 with an offset of 200 ms 30

2.6 Offered Load on Network Topology 1 with an offset of 500 ms 30

2.7 Network Topology 2 31

2.8 Offered Load on Network Topology 2 32

2.9 Total packet drops on Network Topology 2 32

2.10 Offered Load on Link 3-8 33

3.1 Network model-I (NM-I). Each dotted line represents a unicast session. 48

3.2 Network model-II (NM-II). 50

3.3 Network model-III (NM-III). 52

3.4 Network topology 1 62

3.5 Network costs and packet losses with 6 receivers. (a) Network cost, (b) packet losses. 64

3.6 Network costs and packet losses with 11 receivers. (a) Network cost, (b) packet losses. 66

3.7 Network topology 2 68

3.8 Network costs and packet losses - Poisson traffic source. (a) Network cost, (b) packet losses. 70

3.9 Network cost and packet drops with fixed step sizes and decreasing step sizes. 72

3.10 Network cost and packet losses - MMPP source. (a) Network cost, (b) packet losses. 74

4.1 Network topology 1 96

4.2 Variation of maximum link utilization in the first topology under stochastic comparison algorithm. 97

4.3 Network topology 2 99

4.4 Variation of maximum link utilization in the second topology under stochastic comparison algorithm. 99

4.5 Variation of maximum link utilization in the second topology under stochastic comparison algorithm. 101

4.6 Variation of maximum link utilization in the second topology under stochastic comparison algorithm for multicast traffic. 103

Chapter 1

Introduction

Rapid growth of the Internet and the emergence of new demanding services have sparked interests in the Internet traffic engineering. As defined in [4], traffic engineering deals with the issue of performance evaluation and performance optimization of operational IP networks and encompasses the *measurement*, *characterization*, *modeling* and *control* of the Internet traffic.

Due to the evolution of the Internet from ARPANET, traditional routing algorithms for IP networks are mostly based on shortest path routing. However, methods relying on a single path between a source-destination pair cannot efficiently utilize network resources and offer limited control capabilities for traffic engineering [4]. Various solutions derived from shortest path routing algorithms have been suggested, mainly by modifying link metrics in accordance with the network dynamics (See [13, 36]). However, these approaches have several shortcomings that have not been addressed effectively. First, they tend to have network-wide effect and can result in undesirable and unanticipated traffic shifts [4]. Second, these schemes cannot distribute the load among the paths of different cost. Moreover, many of major Internet Service Providers (ISPs) are in various stages of increasing their network capacity and node connectivity [44]. A higher level of network connectivity typically provides multiple paths between source-destination pairs, and offers an opportunity

to better utilize network resources through load balancing that solutions based on shortest path routing cannot make use of.

In this thesis, we focus on the problem of *traffic mapping*, which refers to load balancing the traffic flows along pre-established multiple paths. We provide optimal, yet practical, multipath routing algorithms that can effectively minimize network congestion and maximize the usage of network resource by exploiting the locally collected noisy measurement data.

The paths that the routing algorithms employ can be established in different ways. Multi Protocol Label Switching (MPLS) technology is one method to provide paths as it has the flexibility of creating paths in an explicit manner. However, use of MPLS technology requires the existing IP infrastructure be replaced with MPLS capable devices, and therefore raises a major investment question for the Internet Service Providers (ISPs). In this study, we consider an alternative way to establish multiple paths in an IP network. Specifically, we make use of the overlaying architecture presented in [26], which provides traffic engineering capabilities within a domain without requiring major changes in the infrastructure of IP Networks and addresses some of the limitations of basic shortest path schemes mentioned earlier. This new architecture does not need the traditional IP routers to be replaced or modified. Rather it requires simple devices (such as PCs or network processors) to be carefully placed inside the intra-domain network, creating overlay paths between source-destination (SD) pairs. Furthermore, the architecture allows *gradual deployment* of such devices, resulting in improved network performance with the addition of each new device. This provides ISPs with an alternative solution to

achieve desired level of performance at potentially much lower costs. We adopt this architecture to establish multiple paths between SD pairs on which the traffic load can be distributed effectively. However, our basic load balancing approach can be directly applied to other types of networks with multipath capabilities, such as MPLS based networks.

In the first part of this thesis, we propose a distributed optimal routing algorithm to load balance intra-domain traffic along multiple paths for multiple unicast sources. Our solution is based on stochastic approximation, and makes use of only the local network state information. Our model is similar to that of [11], with the following differences. In [11] the authors mention that the *cost derivatives* cannot be computed and hence, should be estimated by measurements. However, the mathematical analysis given in [11] implicitly assumes that the analytical gradient function is available. In addition, the details on how a cost gradient is estimated are not provided, and the method described in [10] appears to be a variant of a well-known *finite differences* method ([41], [42]). However, the issue of gradient estimate is not clearly or explicitly stated in the aforementioned references. We believe that the gradient estimation method plays a crucial role in the sense that the *convergence* of the optimal routing algorithm strongly depends on the conditions defining this estimation process as described in the stochastic approximation literature (See [42, 43, 14]).

In this study, we consider the same problem (of mapping traffic onto multiple end-to-end paths) while relaxing the assumption that the analytical gradient function is available. We derive our proposed *measurement based algorithm* from the

idea of simultaneous perturbation stochastic approximation (SPSA). SPSA allows us to greatly reduce the number of measurements required for estimating the gradient, while at the same time it achieves approximately the same level of accuracy as the classical finite differences method at each iteration. By reducing the number of measurements, our algorithm achieves faster convergence. This is because each measurement requires a non-negligible amount of time in a networked environment. We will detail these issues of estimation, measurement, and convergence in Chapter 2, and we show in Section 2.5, using simulations, that our SPSA based algorithm outperforms the algorithm proposed in [11].

From a broader perspective, a special case of the proposed algorithm (i.e., single source-destination (SD) case) provides an optimal solution to more general problems that have a *simplex constraint set*. Although applications of SPSA to the constrained optimization problems have generated a certain level of interest in the literature, the simplex constraint set problems have not been handled properly as we will discuss in Section 2.5.

Note that the SPSA algorithm can achieve almost sure convergence, provided that the step size parameter diminishes with the number of iterations. However, such a policy may not be practical under dynamic network conditions as the algorithm may not be able to react to network changes in a timely manner after the step size parameter becomes small. As a result, in practice, we have to reset the step size value after some time interval to ensure that the algorithm is able to react to network dynamics appropriately. An obvious alternative is the use of a constant step size. Even though it is difficult to obtain similar almost sure convergence in a constant

step-size case, it has been shown in [27] that weak convergence (*i.e.,* convergence in distribution) is possible under certain conditions. While the weak convergence is not as strong as almost sure convergence, we demonstrate in Section 2.5 that it does not have a significant effect on the performance of the algorithm in a practical sense.

In the second part of this thesis, we investigate the potential benefits of load balancing both unicast and multicast traffic together within a single domain and propose a practical routing algorithm for carrying out this task using only noisy network measurements.

Multicast traffic over the Internet is growing steadily with increasing number of demanding applications including Internet broadcasting, video conferencing, data stream applications, web-content distributions, and exchange of large data sets by geographically distributed scientists and researchers working in collaboration. Ideally, many of these applications require certain rate guarantees, and providing such guarantees demands that the network be utilized more efficiently than with current approaches to satisfy the rate requirements.

There is a limited amount of existing work on multipath multicast routing. Park and Shin [34] propose a scheme that creates multiple trees between a source and a set of destinations and splits the traffic optimally among the trees. However, the proposed solution covers only a single source case. In addition, it assumes the existence of the gradient of an analytical cost function and that the cost function is strictly convex and continuously differentiable. As discussed in [11], in practice it may be difficult, if not impossible, to precisely define accurate analytical cost

functions due to the dynamic nature of networks. Further, even when an analytical cost function exists, it may not be differentiable everywhere. As we will show in Chapter 2, these assumptions can be relaxed.

In another set of work [33, 48] solutions based on network coding (see [1, 25, 29] for details) are proposed. Even though they approach the problem under a more general architecture, these solutions suffer from the limitations inherited from network coding. First, network coding relies on an unrealistic assumption that the network is lossless as long as the *average* link rates do not exceed the link capacities. In fact, a packet loss can be much more costly when network coding is employed because it can potentially affect the decoding of a large number of other packets. Moreover, similar to earlier efforts, these solutions also assume that there is only one multicast session in the network.

We propose a distributed optimal routing algorithm that balances the load along multiple paths for multiple multicast as well as unicast sessions. As in the unicast only case, our *measurement-based* algorithm does not assume the existence of the gradient of an analytical cost function and is a natural generalization of the *unicast* routing algorithm discussed in Chapter 2. To the best of our knowledge this is the first attempt to address the issue of (optimal) multipath routing with *multiple* multicast sessions in a *distributed* manner, while relying only on (local) network measurements.

In order to study the performance of the proposed algorithm and to understand and quantify the benefits of implementing additional multicast capabilities in the underlying IP network, we consider three different network models with gradually

increasing network capabilities; first, we look at the problem under the traditional network model without any IP multicast functionality. In this model multiple paths are provided using a limited number of (application-layer) overlay nodes that are used as relay nodes. In the second model, we allow IP multicast and load balancing of multicast traffic is carried out utilizing multiple multicast trees available for each source. These trees are rooted either at the source or at the overlay nodes that act as surrogate sources for traffic forwarded by the source. The routers are assumed to be capable of copying and forwarding multicast packets onto downstream branches. Finally, in the third model we replace the routers in the second model with "smart" routers capable of forwarding multicast packets onto each downstream branch at a different rate. This model enables our algorithm to exercise finer control over rate allocation than in the second model. These models together provide us with a general framework that helps us identify functionalities beyond basic operations (e.g., store-and-forward) that are essential for the performance gain from load balancing.

It is worth noting that the finer control available in the third model comes at the price of additional intelligence needed at the routers. This is due to the fact that in order to ensure the delivery of distinct packets to the receivers, a source needs to maintain careful bookkeeping of all the packets forwarded to each receiver so that every packet is forwarded to each receiver and delivery of duplicate packets is minimized. For the same reasons, an intermediate IP router must be able to identify the set of intended receivers for *each* multicast packet. We will show that this problem of potentially complicated bookkeeping at the multicast sources and the core overlay nodes can be avoided by employing a family of source codes called

Digital Fountain codes [31].

Similar to the unicast case, we investigate the convergence performance of our multipath multicast algorithm for both decreasing and fixed step sizes. We first show that under a set of mild conditions the algorithm converges with probability 1 (w. p. 1) to an optimal solution that minimizes the network cost with decreasing step size. Next, for the fixed step size, we show that when the step size is sufficiently small, the algorithm converges to a small neighborhood around the set of optimal points. We demonstrate using simulation results that the performance of a fixed step size algorithm is comparable to that of a policy with decreasing step sizes.

The overlay architecture plays a key role in providing paths for the multipath routing algorithms. By selecting the number, location and the connectivity of the overlay nodes we establish multiple paths between source-destination pairs over which we run our optimal routing algorithms. Clearly, the performance of the routing algorithms are limited by the selection of overlay nodes. Hence, it is of profound importance to optimize the overlay topology to improve the performance of the routing algorithms. In the final part of this thesis, we address the issue of finding good alternative paths by locating a limited number of overlay nodes in the network. To this end, we first formulate a discrete stochastic optimization problem and solve it using the Stochastic Comparison (SC) [16] algorithm. Next, we seek suboptimal solutions suitable for dynamical network environments, stemming from the fact that the SC is only suitable for offline optimization because of its computational complexity. Namely, we develop a *greedy* heuristic solution, which can also allow us gradual deployment of overlay nodes as explained in Chapter 4.

We show through a detailed simulation study that the performance obtained by the heuristic solution is comparable to that of SC algorithm.

Chapter 2

Measurement Based Optimal Multipath Routing for Unicast Sessions

In this chapter, we focus on the assignment of unicast traffic onto pre-established paths to meet certain requirements [4]. We present a distributed optimal routing algorithm based on stochastic approximation theory that uses only noisy local state network measurements.

2.1 Model

We model the network by a set L of unidirectional links. Let $S = \{1, 2, \cdots, S\}$ denote the set of SD pairs. An SD pair s has a set $P_s \subseteq 2^L$ of paths available to it, and $N_s = |P_s|$, *i.e.,* N_s is the number of paths available for SD pair s. Denote the set of links that belong to at least one path $p \in P_s$ by $L^s \subset L$. With a little abuse of notation we let $P_s = \{1, 2, \ldots, N_s\}$, and define the set of all paths $P = \cup_{s \in S} P_s = \{1, 2, \ldots, N\}$, where $N = \sum_{s \in S} N_s$. While by definition, none of the paths can be used by more than one SD pair, any two paths can share a link.

The total input traffic rate of an SD pair s is denoted by r_s, and the SD pair routes x_{sp} amount of traffic on path $p \in P_s$ such that

$$\sum_{p \in P_s} x_{sp} = r_s, \quad \text{for all } s. \tag{2.1}$$

Let $\boldsymbol{x_s} = (x_{sp}, p \in P_s)$ be the rate vector of SD pair s, and let $\boldsymbol{x} = (\boldsymbol{x_s},\ s \in S)$

be the vector of all rates. The flow rate on a link $l \in L$ is given by

$$x^l = \sum_{s \in S} \sum_{p \in P_s \,:\, l \in p} x_{sp} \tag{2.2}$$

For each link l, $C_l(x^l)$ represents the link cost as a function of the link flow rate x^l. We assume that, for all l, $C_l(\cdot)$ is convex and continuously differentiable. The objective is to minimize the total cost $C(\boldsymbol{x}) = \sum_l C_l(x^l)$ by mapping the traffic on paths in P:

$$\min_{\boldsymbol{x}} C(\boldsymbol{x}) = \min_{\boldsymbol{x}} \sum_l C_l(x^l) \tag{2.3}$$

$$\text{s.t.} \sum_{p \in P_s} x_{sp} = r_s, \forall\, s \in S, \tag{2.4}$$

$$x_{sp} \geq \epsilon,\ \forall\, p \in P_s,\ s \in S, \tag{2.5}$$

where ϵ is an arbitrarily small positive constant. For instance, some of the control packets may be routed along different paths available to an SD pair.

Theoretically if the exact gradient values are known, one can use the well known gradient projection algorithm to solve this constrained optimization problem, where the constraint set Θ is defined by (2.4) and (2.5). Each iteration of the algorithm takes the form:

$$\boldsymbol{x}(k+1) = \Pi_\Theta\big[\boldsymbol{x}(k) - a(k)\boldsymbol{\nabla} C(k)\big] \tag{2.6}$$

where $\boldsymbol{\nabla} C(k)$ is the gradient vector whose $(s,p)^{th}$ element is the first derivative length of path $p \in P_s$ at the k-th iteration $\big([\boldsymbol{\nabla} C(k)]_{sp} = \partial C(\boldsymbol{x}(k))/\partial x_{sp}\big)$, $a(k) > 0$ is the step size, and $\Pi_\Theta[\boldsymbol{\vartheta}]$ denotes the projection of a vector $\boldsymbol{\vartheta}$ onto the feasible set Θ with respect to the Euclidean norm.

The above iteration can be carried out in a distributed manner by each pair s without the need to coordinate with other pairs [45], [5]. In other words, the source of each SD pair s updates its rates $\boldsymbol{x}_s$ independently of other SD pairs:

$$\boldsymbol{x_s}(k+1) = \Pi_{\Theta_s}\big[\boldsymbol{x_s}(k) - a_s(k)\boldsymbol{\nabla} C_s(k)\big] \tag{2.7}$$

where $\boldsymbol{\nabla} C_s(k) = (\partial C(\boldsymbol{x}(k))/\partial x_{sp}, p \in P_s)$ is the vector of first derivative lengths of paths in P_s, *i.e.,* a subvector of $\boldsymbol{\nabla} C(k)$ with elements correponding to the paths in P_s, and Π_{Θ_s} denotes a projection onto the feasible set Θ_s of SD pair s.

One problem with directly implementing (2.7) is that the first derivative length of a path, $\partial C/\partial x_{sp}$, may not be available in practice and can only be estimated empirically through *noisy* measurements of the cost function. This is due to the fact that the packet arrival processes are both stochastic and time varying. Therefore, one must resort to a gradient approximation method to obtain an estimate to be used in (2.7). Stochastic approximation methods are natural candidates for such problems.

2.2 Stochastic Approximation

Stochastic Approximation (SA) is a recursive procedure for finding the root(s) of equations using noisy measurements, and is particularly useful for finding extrema of functions [43] (*e.g.,* [24] and [7]).

General constrained SA has the same form as (2.6) with the gradient vector $\boldsymbol{\nabla} C(k)$ replaced by its approximation $\hat{\boldsymbol{g}}(k)$. The approximation is obtained through measurements of $C(\boldsymbol{x})$ around $\boldsymbol{x}(k)$. Under appropriate conditions, one can show

that $\boldsymbol{x}(k)$ converges to the solution set of (2.3), which we denote by $\mathbf{x}^*$.

A critical issue in SA is the approximation of gradient vector. The standard approach motivated from the definition of gradient is the *Finite Differences* (FD) method, in which each component of $\boldsymbol{x}(k)$ is perturbed one at a time and corresponding measurements $y(.)$ are obtained. Typically, the i-th component of $\hat{\boldsymbol{g}}(k)$ ($i = 1, 2, ..., m$) under the FD method is given by

$$\hat{g}_i(k) = \frac{y(\boldsymbol{x}(k) + \xi(k)\boldsymbol{e}_i) - y(\boldsymbol{x}(k) - \xi(k)\boldsymbol{e}_i)}{2\xi(k)}$$

where $\xi(k)$ is some positive number, $\boldsymbol{e}_i$ denotes a unit vector with one in the i-th position and zeros elsewhere, and $y(\cdot)$ denotes the measured cost function with measurement noise.

An alternative method for estimating the gradient is called the *Simultaneous Perturbation* (SP). In this method, all elements of $\boldsymbol{x}(k)$ are randomly perturbed together to obtain two measurements. The i-th component of $\hat{\boldsymbol{g}}(k)$ is computed by

$$\hat{g}_i(k) = \frac{y(\boldsymbol{x}(k) + \xi(k)\boldsymbol{\Delta}(k)) - y(\boldsymbol{x}(k) - \xi(k)\boldsymbol{\Delta}(k))}{2\xi(k)\Delta_i(k)}$$

where the vector of the random perturbations for SP, $\boldsymbol{\Delta}(k) = (\Delta_1(k), \Delta_2(k), \cdots, \Delta_m(k))$, needs to satisfy certain conditions that will be discussed later. Here m denotes the dimension of the vector $\boldsymbol{x}$.

Both of the above approximations have a "two-sided" form in the sense that they use the measurements $y(\boldsymbol{x}(k) \pm \textit{perturbation})$. On the other hand, one-sided gradient approximations require measurements of $y\,(\boldsymbol{x}(k))$ and $y(\boldsymbol{x}(k) + \textit{perturbation})$. Although it is known that the standard two-sided form gives more accurate estimates

compared to one-sided forms, for real-time applications one-sided gradient approximation may be preferred when the underlying system dynamics change too rapidly to get an accurate gradient estimate with two successive measurements [41]. In our work we assume that the one-sided form is utilized for the gradient approximation under both methods unless stated otherwise.

An SA algorithm using the FD (resp. SP) gradient approximation method is referred to as a FDSA (resp. SPSA) algorithm. One should note that, in an SPSA algorithm the gradient approximation requires only two cost function measurements, regardless of the value of m. Standard (two-sided) FD approximation requires $2m$ measurements to estimate the gradient. In [43] it is shown that under reasonably general conditions, SPSA and FDSA achieve the same level of statistical accuracy for a given number of iterations even though SPSA uses m times fewer function evaluations than FDSA. This theoretical result has been confirmed in many numerical studies, even in cases where m is on the order of several hundreds or thousands [41]. This is certainly an important property especially if the measurements are costly and/or time consuming. Clearly, this is the case for the routing problem at hand as measurements require resources and must be collected and reported in a timely manner. In other words, SPSA promises a potential for better statistical accuracy over the same period of "time" due to a much shorter measurement period required at each iteration, even though the two methods have the same statistical accuracy with the same number of "iterations". This suggests that the algorithm based on SPSA will be able to track and respond to changes in a network much faster than another algorithm based on FDSA and thus improve the overall network

performance.

In [43], Spall provides a formal proof of convergence of SPSA algorithm for the "unconstrained" case. The convergence of an SPSA algorithm under inequality constraints are presented in [14, 37]. However, these results do not consider the case where $\boldsymbol{x}(k) \pm \xi(k)\boldsymbol{\Delta}(k) \notin \Theta$, which may occur in our routing problem. In [37] Sadegh suggests projecting $\boldsymbol{x}(k)$ to a point $\boldsymbol{x}^{'}(k) \in \Theta$ such that $\boldsymbol{x}^{'}(k) \pm \xi(k)\boldsymbol{\Delta}(k) \in \Theta$. If $\boldsymbol{x}^{'}(k) - \boldsymbol{x}(k) \rightarrow 0$ as $k \rightarrow \infty$, convergence can still be established. However, when Θ is a simplex, if $\xi(k) \sum_j \Delta_j(k) \neq 0$ then $\boldsymbol{x}^{'}(k) \pm \xi(k)\boldsymbol{\Delta}(k) \notin \Theta$ for all $\boldsymbol{x}^{'}(k)$ (as the demand constraint in (2.4) is violated). Under these conditions, there is no established convergence result of an SPSA algorithm that we can directly apply to our problem. (In [14], although authors claim that they have shown the convergence for the case of a network of queues with similar constraints, they do not consider the aforementioned issue in the proofs.)

In the next section, we will resolve this technical issue by a simple method and present a formal proof of the SPSA algorithm under these constraints.

2.3 Optimal Routing Using SPSA

2.3.1 An Optimal Routing Algorithm - Decreasing Step Size

In this section we propose an optimal routing algorithm and prove its stability and optimality. We know from [45] that if each SD pair runs (2.7) independently

and asynchronously,[1] the overall algorithm converges. Let us now consider the use of SPSA in place of (2.7).

At time k, SD pair s updates its rate according to

$$\boldsymbol{x_s}(k+1) = \Pi_{\Theta_s}[\boldsymbol{x_s}(k) - a_s(k)\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)] \tag{2.8}$$

where $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ is the approximation to $\boldsymbol{\nabla} C_s(k)$ obtained by the SPSA algorithm and is given by

$$\begin{aligned}\hat{g}_{s,i}(k) :&= \frac{N_s}{N_s - 1}\frac{y_s(\Pi_\Theta[\boldsymbol{x}(k) + \boldsymbol{\Xi}(k)\boldsymbol{\Delta}(k)]) - y_s(\boldsymbol{x}(k))}{\xi_s(k)\Delta_{s,i}(k)} \\ &= \frac{N_s}{N_s - 1}\frac{(C_s^+(k) + \mu_s^+(k)) - (C_s^-(k) + \mu_s^-(k))}{\xi_s(k)\Delta_{s,i}(k)}\ , \\ &\qquad i = 1, \ldots, N_s,\end{aligned} \tag{2.9}$$

Here $\boldsymbol{\Delta}(k) = (\boldsymbol{\Delta_s}(k),\ s \in S)$, $\boldsymbol{\Delta_s}(k)$ is the random perturbation vector for source s at iteration k, $\boldsymbol{\Xi}(k)$ is an $N \times N$ diagonal matrix whose j-th diagonal entry is equal to ξ_{s_j} (s_j being the SD pair associated with the j-th component of $\boldsymbol{\Delta}(k)$), and $y_s(\cdot)$ are the noisy measurements of the *partial* cost information, which is the summation of the link costs over the set L^s. Specifically, $C_s^-(k) = \Lambda_s(\boldsymbol{x}(k)) := \sum_{l \in L^s} C_l(x^l(k))$ and $C_s^+(k) = \Lambda_s(\Pi_\Theta[\boldsymbol{x}(k) + \boldsymbol{\Xi}(k)\boldsymbol{\Delta}(k)])$ with $\mu_s^+(k)$ and $\mu_s^-(k)$ being the measurement noise terms.

Note that (2.9) differs from the standard SA in the following ways. First, each SD pair uses only *partial* cost information (*i.e.,* summation of the costs of the links in L^s) as opposed to the total network cost, which is the summation of the costs of all the links in the network. In addition, the noise terms observed by each SD

[1]Here asynchrony refers to the fact there might be a time lag between SD pairs in continuous time operation in practice.

pair are allowed to be different. Second, while $\xi(k)$ is a positive scalar in standard SA, in our case $\boldsymbol{\Xi}(k)$ is an $N \times N$ diagonal matrix. This allows the possibility of having different $\xi_s(k)$ values for different SD pairs. Also, note that we have an extra multiplicative factor $\frac{N_s}{N_s-1}$ in (2.9) compared to the standard SA. This term is required for convergence as an outcome of the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ to Θ_s for all $s \in S$ using L_2 projection while calculating $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$. Finally, if $\Pi_{\Theta_s}[\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)] = \boldsymbol{x_s}(k)$, the SD pair draws a new perturbation vector $\boldsymbol{\Delta_s}(k)$ until $\boldsymbol{x_s}(k) \neq \Pi_{\Theta_s}[\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)]$.

Note from (2.8) that SD pairs may have different step sizes $a_s(k)$. This allows a certain level of asynchrony between SD pairs in the sense that SD pairs can respond to the network changes independently of each other to some extent.For instance, this formulation covers the case where SD pairs start the algorithm at different times. However, we assume that SD pairs update their rates once every iteration after they start the algorithm. This assumption is reasonable in our case, since at each iteration SD pairs should make use of the collected information that is already available. This is, however, not to say that the updates take place simultaneously. The errors due to this asynchrony are assumed to be absorbed into the error terms $\mu_s^{-+}(k)$ in (2.9).

For the optimality of the new algorithm, we need to show that (2.8) converges to the same point $\boldsymbol{x}_{\boldsymbol{s}}^*$ as (2.7) for all SD pairs. For the convergence of the algorithm we assume that the following conditions are true:

A1. $C(\boldsymbol{x}(k))$ is differentiable for each $\boldsymbol{x}(k) \in \Theta$, and convex.

A2. The perturbation terms $\Delta_{s,i}(k)$ are (i) mutually independent with zero mean for all $s \in S$ and $i \in P_s$, (ii) uniformly bounded by some constant $\alpha < \infty$ with support on finite discrete sets, (iii) independent of $(\boldsymbol{x}(n), n = 0, 1, \cdots, k)$, and (iv) $\mathbf{E}\left[(\Delta_{s,i}(k))^{-1}\right]$, $\mathbf{E}\left[(\Delta_{s,i}(k))^{-2}\right]$ are bounded for all k.

A3. $\mathbf{E}\left[\mu_s^{(\pm)^2}(k)|\boldsymbol{\Delta}(k), \mathcal{F}_k\right]$ are bounded and $\mathbf{E}\left[\mu_s^+(k) - \mu_s^-(k)|\boldsymbol{\Delta}(k), \mathcal{F}_k\right] = 0$ a.s. for all k, where $\mathcal{F}_k$ is the σ-field generated by $\{\boldsymbol{x}(0), \cdots, \boldsymbol{x}(k)\}$.

A4. (i) $\sum_{k=0}^{\infty} a_s(k) = \infty$, (ii) $a_s(k) \to 0$ as $k \to \infty$, (iii) $\sum_{k=0}^{\infty} \left(\frac{a_s(k)}{\xi_s(k)}\right)^2 < \infty$, (iv) $\xi_s(k) \to 0$ as $k \to \infty$, and (v) $\lim_{k\to\infty} \left(\frac{\xi_s(k)}{\xi_{s'}(k)}\right) = 1$ for all $s, s' \in \mathcal{S}$.

A5. Define $\hat{a}(k) := \max_{s\in S} a_s(k)$. (i) $\sum_{k=0}^{\infty}(\hat{a}(k) - a_s(k)) < \infty$ for all $s \in \mathcal{S}$, and (ii) $\lim_{k\to\infty} \frac{a_s(k)}{\hat{a}(k)} = 1$.

Theorem 2.3.1. *Under Assumptions **A1**-**A5**, the sequence $\boldsymbol{x}(k) = (\boldsymbol{x_s}(k), s \in S)$ generated by the algorithm defined by* (2.8) *converges to $\mathbf{x}^*$ with probability 1, where $\mathbf{x}^*$ is the solution set of the optimization problem in* (2.3)*, regardless of the initial vector $(\boldsymbol{x_s}(0), s \in S) \in \Theta$.*

Proof: The proof of Theorem 2.3.1 is given in Section 2.7. □

2.3.2 The Optimal Routing Algorithm - Constant Step Size

In the previous subsection we have employed decreasing step sizes $a_s(k)$, $k = 0, 1, \ldots$, as defined in **A4** and **A5**. Although it is possible to use decreasing step

sizes, it is of practical importance to consider the case with constant step size $a > 0$ given in the following form

$$\boldsymbol{x_s}(k+1) = \Pi_{\Theta_s}[\boldsymbol{x_s}(k) - a \cdot \hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)]. \tag{2.10}$$

This is because in the case of decreasing step sizes, one has to make sure that each source node resets the step size after a certain period of time when the step size becomes too small to effectively react to the dynamics of the network. Consequently, such a requirement introduces additional complexity which can be avoided by using a constant step size. The main difficulty with constant step size algorithms is that it is difficult to establish almost sure convergence. However, as shown in [27], under certain conditions constant step size SA algorithms can achieve weak convergence (*i.e.,* convergence in distribution), which can be interpreted as convergence to a neighborhood of the optimal operating point(s). Since the performance of the system near the optimal operating point(s) may be comparable to that of the optimal solution(s) in a network problem, the performance degradation, if there is any, due to a constant step size may not be significant. This is supported by simulation results in Section 2.5.

The following theorem establishes the (weak) convergence of the algorithm with a constant step size.

Theorem 2.3.2. *Under Assumptions **A1**-**A3**, for any $\delta > 0$, the fraction of time the sequence $\boldsymbol{x}(k) = (\boldsymbol{x_s}(k), s \in S)$ generated by the algorithm defined by* (2.10) *spends in δ-neighborhood of $\mathbf{x}^*$ on $[0, k]$, goes to one (in probability) as $a \to 0$ and $k \to \infty$ regardless of the initial vector $(\boldsymbol{x_s}(0), s \in S) \in \Theta$.*

Proof: The proof of Theorem 2.3.2 is given in Section 2.8. □

2.3.3 Measurement Process

In this section, we discuss some of the issues regarding the measurement process and their effect on the overall performance of the algorithm. We will also point out the benefits of SPSA based algorithms over the FDSA alternatives.

By definition, an FDSA based algorithm requires SD pair s to perturb its paths *one at a time*, requiring $N_s + 1$ measurements for the one-sided form and $2N_s$ measurements for the two-sided form in each iteration for the estimation of an $N_s \times 1$ gradient vector. For this reason, FDSA based algorithms need each SD pair to collect measurements (*i.e.,* perturb its paths) at different times. As mentioned in [11], this requires a coordination protocol that determines the order in which paths are perturbed. Moreover, it increases network traffic load by creating extra communication overhead.

SPSA, on the other hand, allows us to estimate the gradient vector using only two measurements. In the context of our routing problem, this implies that an SD pair can perturb all of its paths simultaneously if an SPSA based algorithm is employed. In addition, it also suggests that all SD pairs can execute the perturbation in parallel without the need for strict coordination needed in FDSA (Theorem 2.3.1 and Theorem 2.3.2). This enables SD pairs to operate independent of each other to some extent. As a consequence, a potential overhead that would be incurred by a coordination protocol under an FDSA algorithm is avoided. Furthermore, we can

significantly reduce the duration of measurement periods by concurrently executing the measurement process at the SD pairs. Since the statistical accuracy of SPSA does not degrade from that of FDSA, we can achieve much faster convergence due to the fact that we significantly reduce the time required for each iteration.

Another issue regarding the measurement process is the effect of asynchronous operation of SD pairs in practice due to the lack of perfect synchronization. It is proved in [11] that, with increasing level of asynchrony, the convergence becomes slower. For each $s \in S$, let t_0^s be the time lag between the time traffic measurements are collected and the time SD pair s carries out a rate update. In other words, t_0^s is the delay between the collection of measurements and execution of an SPSA algorithm by SD pair s. Define $t_0 = \max_{s \in S} t_0^s$. Then, the larger the t_0 is, the slower the convergence will be due to the use of delayed information. On the other hand, in the SPSA case as the level of asynchrony between the SD pairs increases, on the average, the magnitude of the error term in measurements tends to become smaller since the duration of the interval over which the measurements overlap with each other gets shorter, and this may cause a marginal increase in the overall system performance. As we will see in Section 2.5, these two effects mainly cancel each other and the performance of the algorithm does not degrade significantly if the time lag is not large.

2.4 Implementation Issues

In this section, we give a brief overview of an overlay architecture used to enable traffic engineering capabilities. The details of the overlay architecture can be

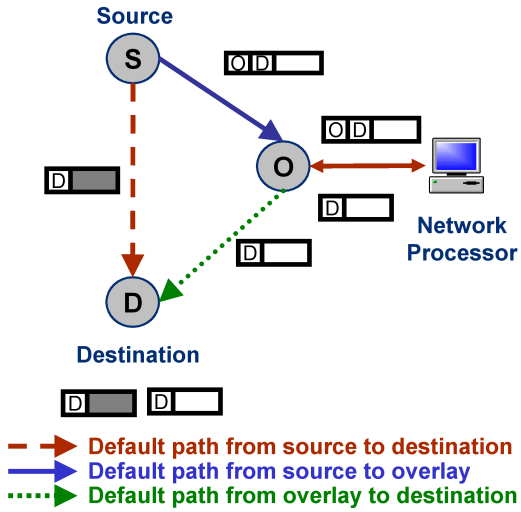


Figure 2.1: Proposed overlay architecture for multipath routing.

found in [26].

2.4.1 Path Establishment

Alternative paths between SD pairs are created using overlay nodes. The overlay nodes are located at all source-destination nodes as well as at some core nodes. The basic idea is similar to the ones presented in [9] and [2], with the difference being that the overlay is implemented within a domain (*i.e.,* intra-domain) as opposed to inter-domain. When a packet is sent along the default path (*e.g.,* the shortest path), it is forwarded in the same way as in traditional IP networks. However, when a packet is to be routed through an alternative path, it is processed at the source overlay node and an additional IP header is attached to the packet. This way the packet is first tunneled to a specific overlay node that lies in the selected alternative path, using the underlying routing protocol in use. This is shown in Figure 2.1.

When the overlay node receives the packet, it removes the outer IP header and forwards the packet to the final destination (or possibly to another overlay node). It is plain to see that one can utilize as many alternative paths as needed by placing overlay nodes at appropriate locations. Note that using this architecture, we can still employ the simple shortest path routing inside the network, without having to modify the existing traditional routers. The overlay capabilities can be realized by attaching a simple device (*e.g.,* a host with network processor) to the existing routers. This device simply processes the packets, and adds or removes IP headers before the basic forwarding operation is performed at the routers.

As a final remark, we would like to emphasize the point that the basic mechanism of the proposed routing algorithm can be employed in different types of networks. For instance, it can also be deployed in an MPLS based network, where the overlay paths are replaced with LSPs (Label Switched Paths). The proposed use of overlay architecture enables us to adopt the algorithm in the traditional IP-based networks.

2.4.2 Traffic Monitoring

Traffic monitoring is also handled by the overlay architecture. Each link in the network is mapped to the closest overlay node with a tie-breaking rule that gives a unique mapping [26]. Overlay nodes periodically poll the links that they are responsible for, process the data and forward necessary local state information to the SD pairs utilizing the corresponding links. This eliminates the need for *each* SD pair to probe the links used by it. Note that, before forwarding the link

cost information to source nodes of SD pairs, the overlay nodes can aggregate the information gathered from different links. For example, due to the additive cost structure (according to the definition given in (2.3)), if the overlay nodes are aware of L^s, an overlay node can first compute the sum of the link costs over the links in L^s it is responsible for and then report only the total cost to the source node of SD pair s. As a consequence, the overhead caused by the distribution of the link state information is minimized.

2.5 Experimental Setup and Simulation Results

In this section we evaluate the performance of the proposed algorithms through simulations under various network conditions, and study their convergence rates. To this end, we developed a packet level discrete-event simulator to carry out the simulations. Each plot presented in this section is the average of 10 independent simulation runs with different random seeds.

For the simulations we select the cost function of the form

$$C_l(\boldsymbol{x}) = d_l(\boldsymbol{x}) + u_l(\boldsymbol{x})^2 \tag{2.11}$$

where $d_l(\boldsymbol{x})$ is the number of packets dropped on link l,[2] and $u_l(\boldsymbol{x})$ is the link utilization. In all simulations, the measurement period is selected to be one second. As a consequence, SD pairs can update their rates at best approximately every

[2]In the simulation we have used the observed number of packet drops, while in the model this term in the cost function can be interpreted as the expected number of packet losses as a function of the link load.

2 seconds since they require two measurements for estimating the gradient vector using the SPSA algorithm.

The requirements on the cost function are stated in Assumption **A1**, and one can argue that the selected cost function satisfies these requirements as follows. Since we deal with backbone networks, the packet arrival process at a source node is an aggregate of many individual flows. We assume that each individual flow generates packets according to an equilibrium renewal process, *i.e.,* interarrival times of packets from a flow have a fixed distribution, and these equilibrium renewal processes are mutually independent. Then, by the Palm-Khintchine theorem [22], the superposition of these independent renewal processes can be approximated by a Poisson process, where interarrival times of packets are exponentially distributed.

In addition, according to the work presented in [8], the packet size distribution of Internet traffic has two peaks at 500 and 1,500 bytes. Using this observation, we can approximate the packet size as a Bernoulli random variable with values at 500 and 1,500 bytes.

Under the above conditions, we can approximate the links in the network as $M/G/1/K$ queues, where K is the buffer size. Following this assumption we can justify the assumption on convexity of the cost function as follows. One can check that in the regime of interest (*e.g.,* with utilization level being less than 150 percent), the link cost function is convex in the case of $M/M/1/K$ queue. In the case of $M/G/1/K$ queue one can show that the approximation functions for blocking probability of an $M/G/1/K$ queue (*e.g.,* Gelenbe's formula [15] and two-moment approximation in [40]), are indeed convex in the regime of interest under various

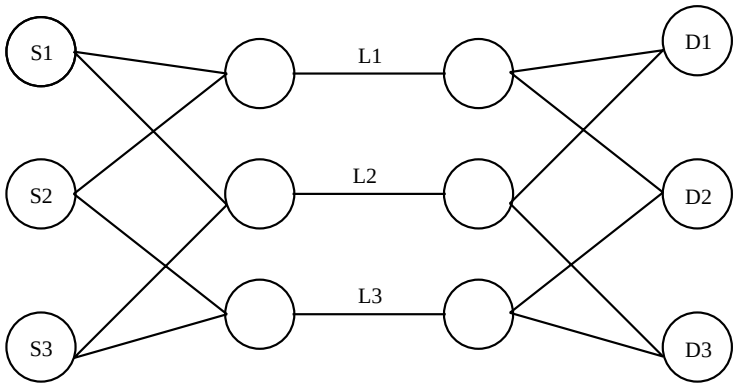


Figure 2.2: Network Topology 1

parameter settings.

Simulations are carried out under two different network topologies. The first topology, which is used in [11], is given in Figure 2.2. Due to its simplicity this topology allows us to obtain insights into the fundamental behavior of the proposed algorithm. In addition, it serves us as a base setup so that we can make a comparison with the MATE algorithm presented in [11]. We have three SD pairs (S1-D1, S2-D2 and S3-D3) and each pair has two distinct paths. Note that links L1, L2 and L3 are shared by paths of all the three SD pairs, which creates a considerable amount of interaction between these SD pairs.

The setup we use is similar to the one adopted in [11] for comparison purposes. (See [38] for the details of this setup.) The network consists of identical links with a bandwidth of 45 Mbps. The average packet size is 257 bytes. Each SD pair initially uses only the default shortest (minimum hop distance) path. Since all paths have equal length, the default min-hop paths are selected such that L2 is along the default shortest path of S1-D1, while the default shortest paths of S2-D2 and S3-D3 both

traverse L3. Each SD pair generates traffic according to a Poisson process with an average rate of 19.8 Mbps (corresponding to 0.44 link utilization). In addition, links L1, L2 and L3 carry uncontrolled cross traffic generated by Poisson processes. The average rate of cross traffic normalized by the link capacity is given in Table 2.1. A random delay is introduced before each SD pair starts running the optimal routing algorithm to ensure that the SD pairs are not synchronized. (The maximum value of this random delay is defined as offset.) In this simulation a decreasing step size policy is adopted that satisfies the conditions in Assumptions **A3**-**A6**. Specifically, the step sizes are given by $a_s(k) = 15/(k+100)^{0.602}$ and $\xi_s(k) = 75/(k^{0.101})$.[3]

	Load Distribution in Time (sec)		
Link	[0-1000)	[1000-2500)	[2500-3600)
$L1$	0.77	0.44	0.44
$L2$	0.33	0.33	0.67
$L3$	0.33	0.33	0.33

Table 2.1: The Cross Traffic Dynamics

As shown in Figs. 2.3 and 2.4, the algorithm quickly eliminates the congestion and successfully balances the traffic. Moreover, these results show that the proposed algorithm clearly outperforms the MATE algorithm [11]; while the MATE algorithm requires around 400-500 seconds to converge,[4] our algorithm takes around 200 sec-

[3] $a_s(k)$ and $\xi_s(k)$ are reset to their initial values at simulation times 1,001 and 2,501 seconds for faster convergence.

[4] Since simulation code and packet size distributions for the MATE algorithm are proprietary, it was not possible to simulate MATE. Therefore, we base our comparison on the results presented in [11].

onds. Furthermore, our algorithm quickly (around 50 seconds) eliminates packet drops unlike MATE. (See Figs. 10 and 11 presented in [11].)

Figs. 2.5 and 2.6 illustrate the effect of increased asynchrony between SD pairs. We increase the asynchrony between SD pairs by simply increasing the offset values. From these plots we can conclude that the algorithm is still able to converge in a short time. As we see from Figs. 2.3 and 2.5, the performance is almost the same for offset values of 50 ms and 200 ms. However, when we increase the offset to 500 ms, we see that the convergence of our algorithm becomes slightly slower. Thus, these results validate the earlier discussion made in subsection 2.3.3.

Figure 2.7 represents the second topology we consider. This topology is also used in [32], [3] and closely resembles the MCI Internet topology presented in [30]. Using this topology, we analyze the performance of the proposed algorithm under more realistic network conditions.

Nodes 1, 5, 6, 14 and 18 serve as both source and destination nodes. This gives us a total of 20 SD pairs. Each pair has at least two paths to reach the destination. A total of 78 paths are created between these 20 SD pairs, using overlay architecture. Overlay capability is available at all source/destination nodes as well as nodes 2, 10 and 13. In this experiment, the offset is set to 0.1 sec. The links shown as dashed lines have a capacity of 50 Mbps, while the links represented by solid lines have a capacity of 20 Mbps. The packet size for this scenario is selected to be 500 bytes. All SD pairs initially use only the shortest paths. Each SD pair generates traffic with a rate of 11.5 Mbps. In addition, the cross traffic traverses link (3-12), starting at simulation time 1,600 sec. The cross traffic rate is 18 Mbps and cannot be shifted

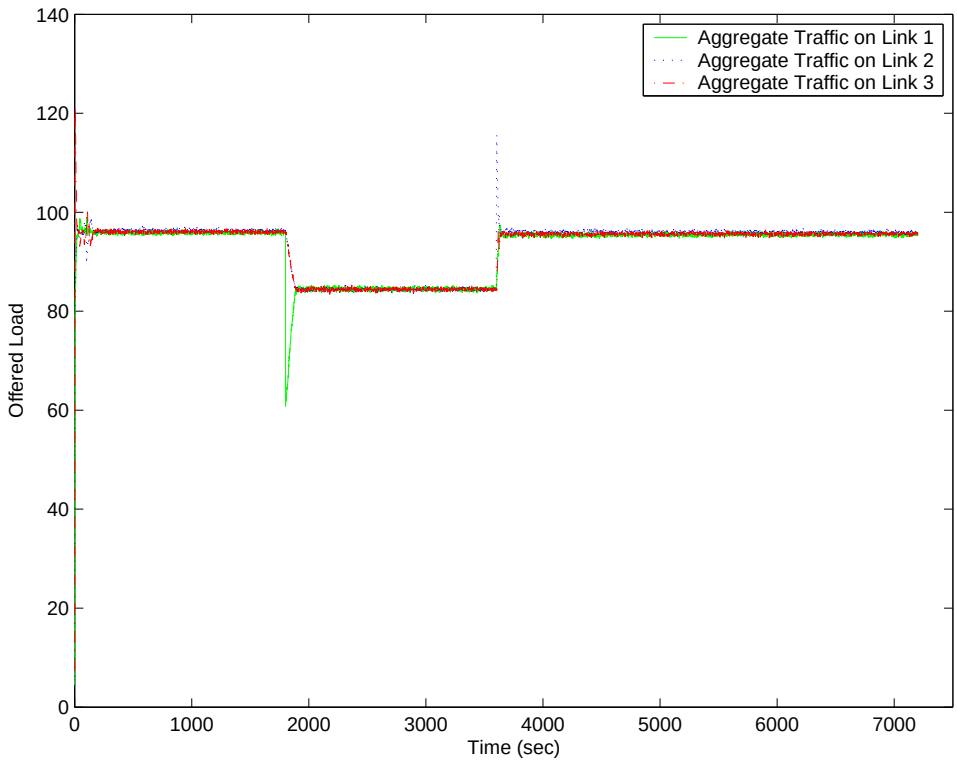


Figure 2.3: Offered Load on Network Topology 1 with an offset of 50 ms

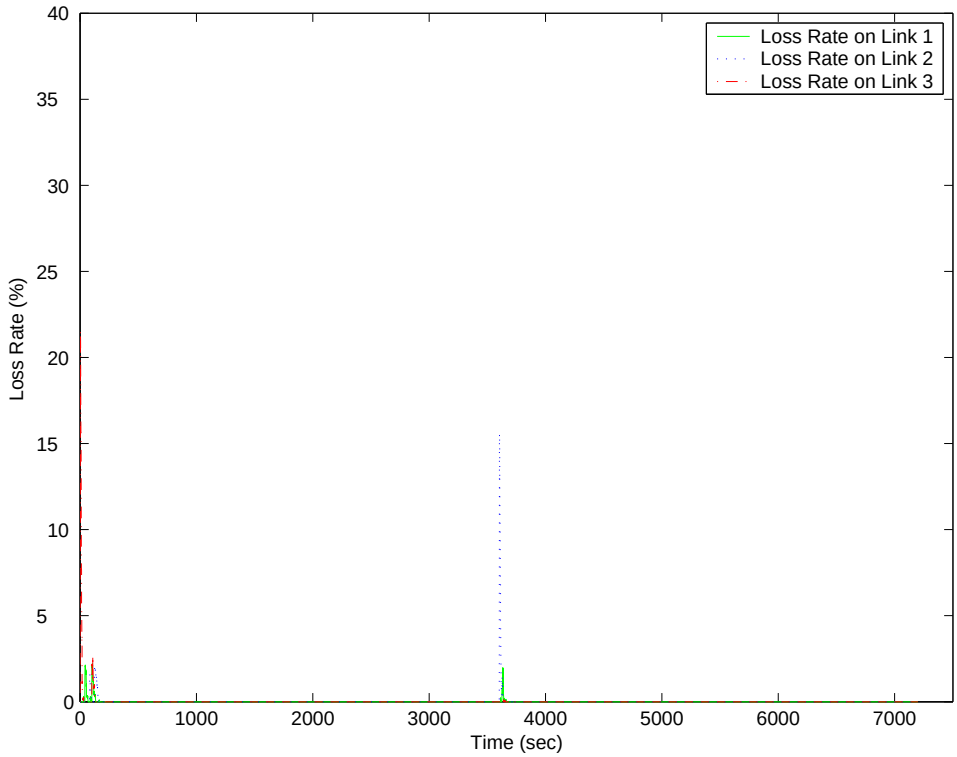


Figure 2.4: Total packet drops on Network Topology 1 with an offset of 50 ms

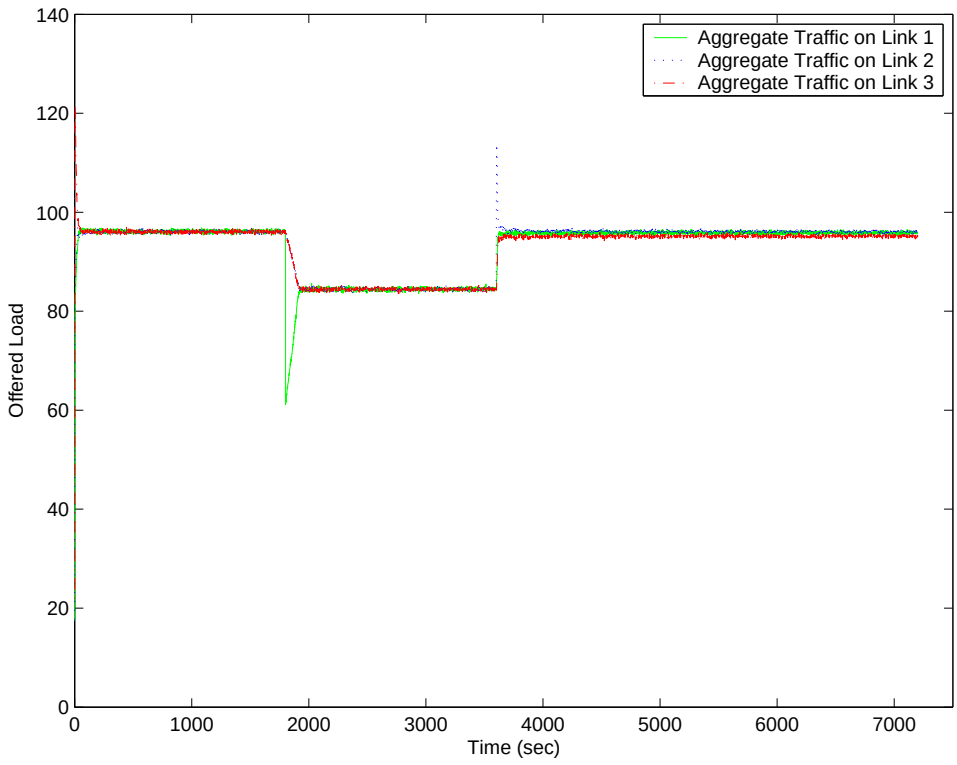


Figure 2.5: Offered Load on Network Topology 1 with an offset of 200 ms

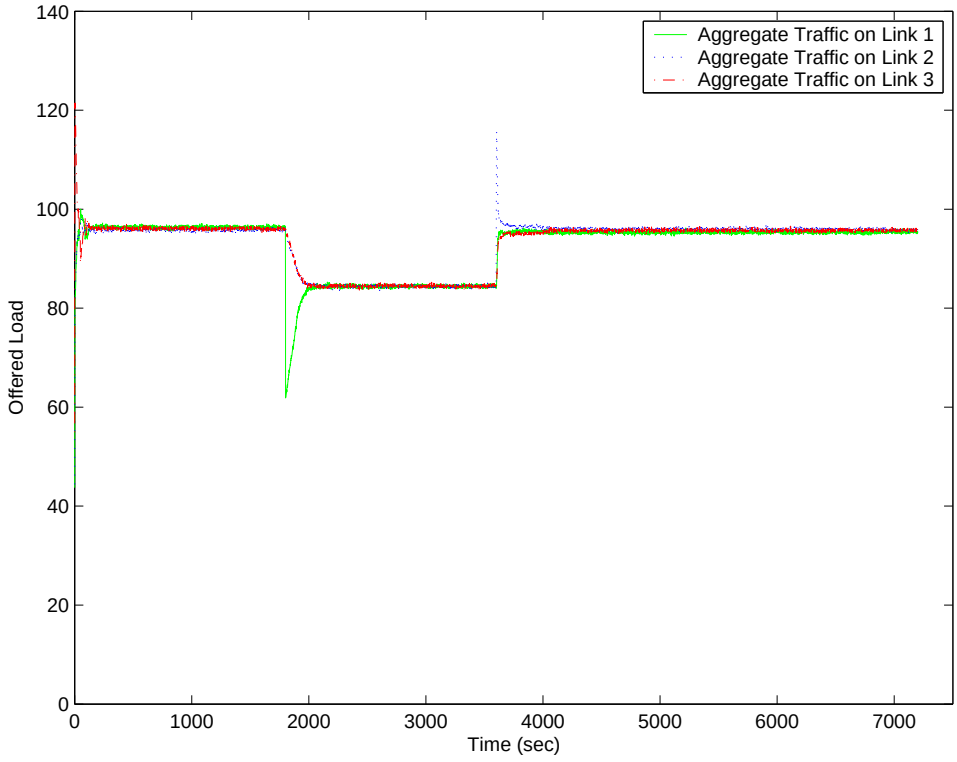


Figure 2.6: Offered Load on Network Topology 1 with an offset of 500 ms

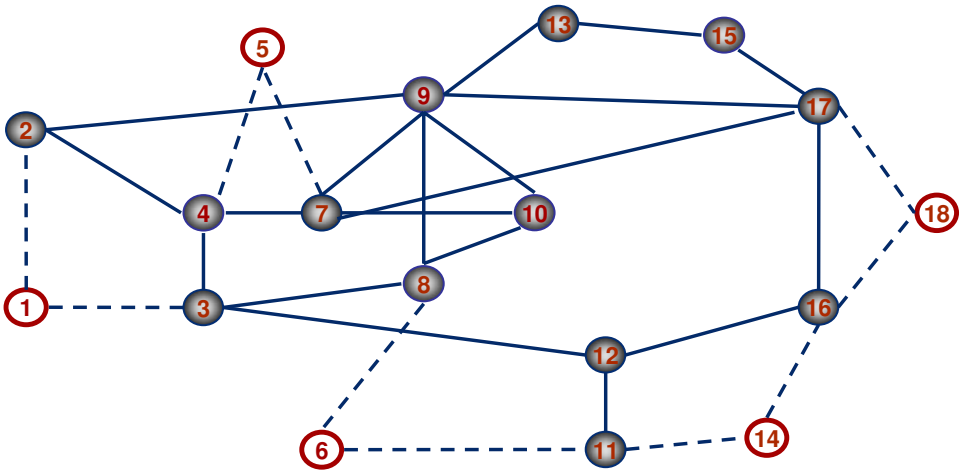


Figure 2.7: Network Topology 2

to any alternative paths as before.

In Figure 2.8, we illustrate how the load is distributed by our algorithm.[5] The links for which we plotted the load are selected in such a way that each of them is located on a distinct alternative path that can be used to divert the traffic sent on link (3-12). The only exception is link (12-16), which is a downstream link of link (3-12) for some of the paths utilized by SD pairs. The plot of traffic on link (12-16) shows how the traffic load is migrated away from the paths that traverse link (3-12). In addition, Figure 2.9 shows the total number of packets dropped in the entire network. We observe from both figures that the algorithm quickly removes congestion as the packet drops are eliminated fast and distributes the load among the multiple paths between the SD pairs in a reasonable amount of time. This result indicates that the proposed algorithm successfully converges under the scenarios where many SD pairs operate in an independent and asynchronous fashion.

[5] $a_s(k)$ is reset to its initial value (0.15) at simulation time 1601 seconds for faster convergence.

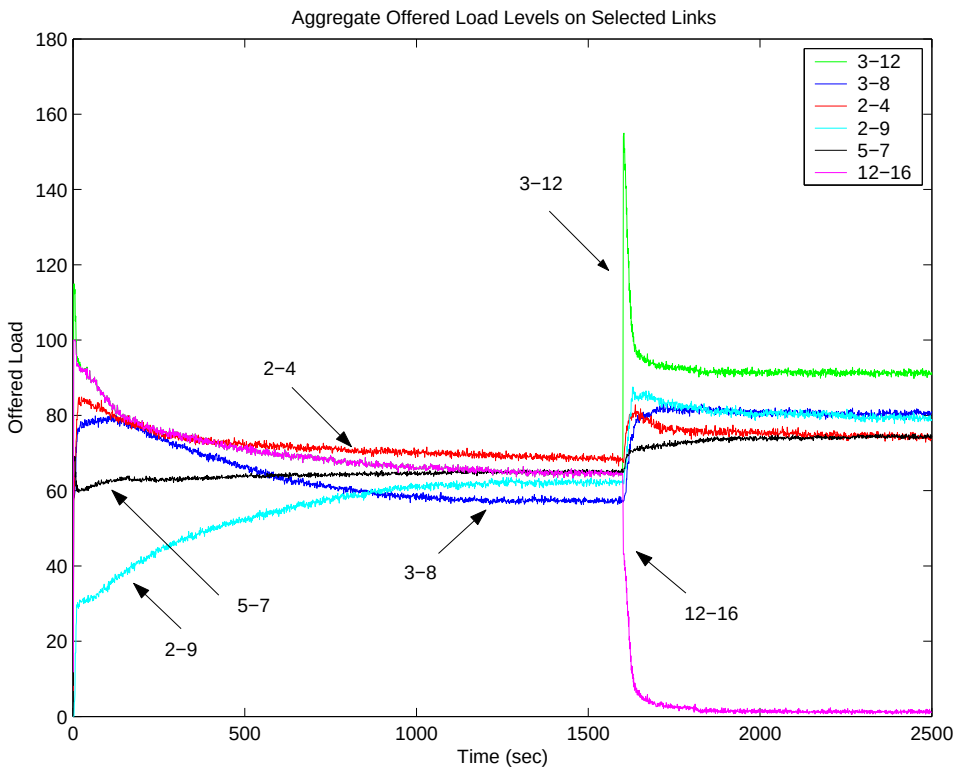


Figure 2.8: Offered Load on Network Topology 2

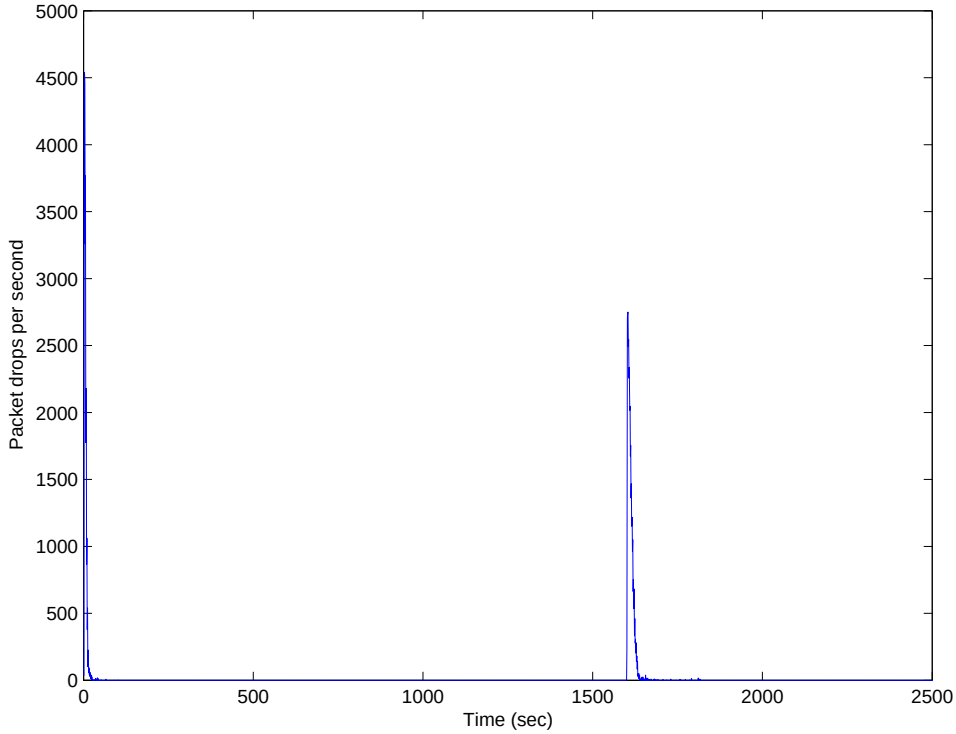


Figure 2.9: Total packet drops on Network Topology 2

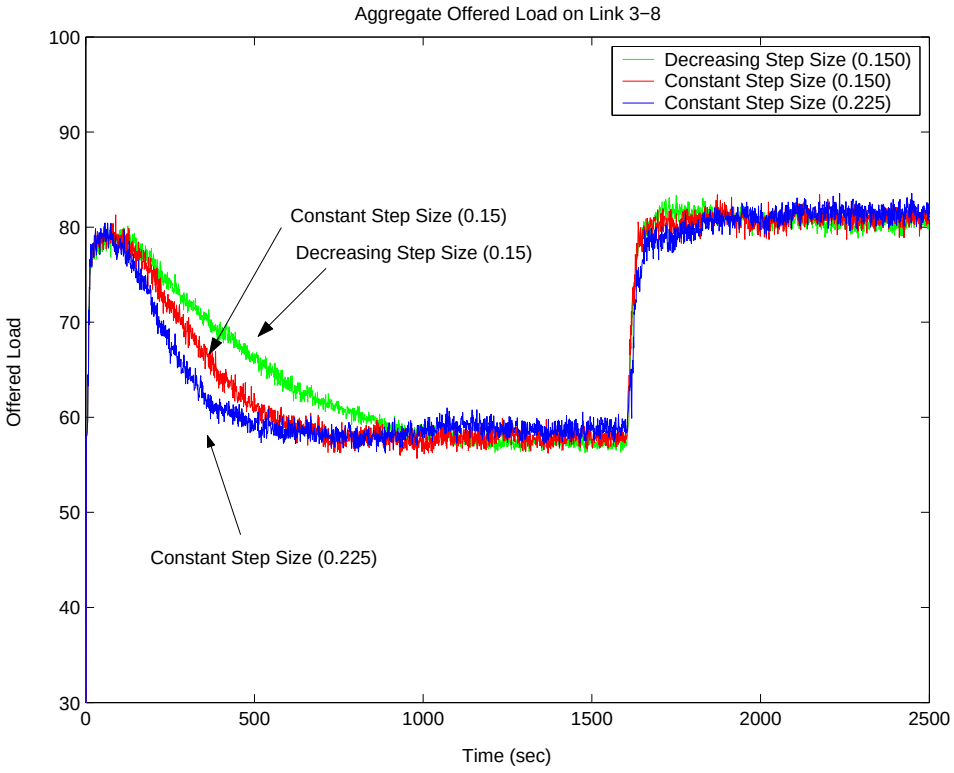


Figure 2.10: Offered Load on Link 3-8

Figure 2.10 demonstrates the effect of using a constant step size. We plot the variation of traffic load on link (3-8) using two different values of fixed step size $a_s(k) = a = 0.15$ and $a_s(k) = a = 0.225$, and compare their performance with the decreasing step size case where the step size for each SD pair is given by $a_s(k) = 15/(k+100)^{0.602}$. Even though we do not have almost sure convergence with constant step size, our results suggest that the convergence in distribution is sufficient for practical purposes. Moreover, the initial convergence rate improves slightly as the algorithm is able to reach a small neighborhood of the optimal operating point faster compared to the decreasing step size case. Since this neighborhood appears to be small and a constant step size policy gives us the robustness to track dynamical changes in the network, this result supports the use of a constant step size algorithm in a practical implementation.

2.6 Conclusion

In this chapter, we have considered optimal multi-path routing in environments where the link cost derivatives can be estimated (but for which an analytic expression may not exist). We have mathematically proven the optimality and stability of our proposed scheme, which is based on simultaneous perturbations. We have demonstrated that an SPSA algorithm provides significant improvements over an algorithm based on traditional finite-difference methods. Specifically, we have shown that our scheme results in much shorter measurement periods during the gradient estimation phase, and as a result, converges faster. Our simulation results

show that our scheme can quickly alleviate network congestion and distribute load efficiently under dynamic network conditions.

The results of this chapter have been published in [18, 19].

2.7 Proof of Theorem 2.3.1

Collecting the terms of (2.8) for all SD pairs we have

$$\boldsymbol{x}(k+1) = \Pi_{\Theta}[\boldsymbol{x}(k) - \boldsymbol{A}(k)\hat{\boldsymbol{g}}(k)], \tag{2.12}$$

where $\hat{\boldsymbol{g}}(k) := (\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k), s \in S)$, and $\boldsymbol{A}(k)$ is an $N \times N$ diagonal matrix and the diagonal entries of $\boldsymbol{A}(k)$ are given by the corresponding step sizes of SD pairs, $a_s(k)$. This is different from the general stochastic approximation algorithms in the sense that the step size $a(k)$ is no longer a scalar. Following [28, Theorem 5.3.1] closely, we will utilize the so-called "ODE method" to show convergence. We rewrite (2.12) in the following form

$$\boldsymbol{x}(k+1) = \boldsymbol{v}(k) + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k) \tag{2.13}$$

where

$$\begin{aligned}
\boldsymbol{v}(k) &= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{\nabla}C(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)], \\
\boldsymbol{\varrho}(k) &= \mathbf{E}\left[\hat{\boldsymbol{g}}(k)|\boldsymbol{x}(k)\right] - \hat{\boldsymbol{g}}(k), \\
\boldsymbol{b}(k) &= \boldsymbol{\nabla}C(\boldsymbol{x}(k)) - \mathbf{E}\left[\hat{\boldsymbol{g}}(k)|\boldsymbol{x}(k)\right], \\
\boldsymbol{\tau}(k) &= \Pi_{\Theta}[\boldsymbol{v}^{\gamma}(k)] - \boldsymbol{v}^{\gamma}(k), \\
\boldsymbol{v}^{\gamma}(k) &= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{\nabla}C(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)]I(k), \\
\boldsymbol{\phi}(k) &= \boldsymbol{v}^{\gamma}(k) - \boldsymbol{v}(k) + \big(\Pi_{\Theta}[\boldsymbol{v}(k)] - \boldsymbol{x}(k)\big)\big(1 - I(k)\big),
\end{aligned}$$

$I(k)$ denotes the indicator function of the event $|| \boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \leq \gamma(k)/2$, and $\gamma(k)$ is a sequence of positive real numbers such that (i) $\gamma(k) \to 0$ and (ii) $|| \boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \leq \gamma(k)/2$ for all but a finite number of k w.p. 1. The following lemma guarantees the existence of such a sequence.

Lemma 2.7.1. *Under the assumptions* ***A1-A4****, for the SPSA gradient estimator* $\hat{\boldsymbol{g}}(k)$ *defined in* (2.9)*, the bias and error terms given by* $\boldsymbol{b}(k)$ *and* $\boldsymbol{\varrho}(k)$*, respectively, satisfy*

(a) $\boldsymbol{b}(k) \to 0$ *w.p. 1,*

(b) $\sum_{k=0}^{\infty} \mathbf{E}\left[|| \boldsymbol{A}(k)\boldsymbol{\varrho}(k)||^2\right] < \infty$ *w.p. 1.*

Proof: Given any $N_s \times 1$ vector $\boldsymbol{\vartheta}$, the solution of the minimization problem

$$\begin{aligned} &\min_{\boldsymbol{\eta}} ||\boldsymbol{\vartheta} - \boldsymbol{\eta}||^2 \\ &\text{s. t. } \boldsymbol{\eta}^T \boldsymbol{u} = r_s \end{aligned} \tag{2.14}$$

is given by

$$\eta_i = \vartheta_i + \frac{r_s - \sum_{j=1}^{N_s} \vartheta_j}{N_s} \ , \tag{2.15}$$

where $\boldsymbol{u} = [1, 1, \cdots, 1]^T$. Obviously, if $\eta_i \geq 0$ for all i, this solution is equivalent to the L_2 projection. Here for the purpose of temporary perturbation we replace (2.5) with a non-negativity constraint. Thus, the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ can be calculated using (2.15) if

$$x_{s,i}(k) + \xi_s(k)\left(\Delta_{s,i}(k) - \frac{\sum_{j=1}^{N_s}(\Delta_{s,j}(k))}{N_s}\right) \geq 0 \ . \tag{2.16}$$

Recall that $\Delta_{s,i}(k)$ is bounded by α from Assumption **A2**. Hence, (2.16) holds if

$$\xi_s(k) \leq \frac{\min_j\{x_{s,j}(k)\}}{2\alpha} \ . \tag{2.17}$$

From (2.5) we know $\frac{\min_j\{x_{s,j}(k)\}}{2\alpha} \geq \frac{\epsilon}{2\alpha}$. Since, $\xi_s(k) \to 0$, there exists finite K_1 such that $\xi_s(k) \leq \frac{\epsilon}{2\alpha}$ for all $k > K_1$. Therefore, (2.15) can be used to compute the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ for sufficiently large $k > K_1$.

Let $\bar{\boldsymbol{\Delta}}_{\boldsymbol{s}}(k)$ be an $N \times 1$ vector, where values of entries corresponding to those of SD pair s are $\Delta_{s,i}(k)$ and zero otherwise. Hence, $\sum_{s\in S} \bar{\boldsymbol{\Delta}}_{\boldsymbol{s}}(k) = (\Delta_{s,i}, s \in S, i \in P_s)$. Similarly, $\boldsymbol{u_s}$ is an $N \times 1$ vector, where the values of entries corresponding to those of SD pair s are one and the remaining entries are zero. Following the proof in [14] and using Taylor's theorem, for $k > K_1$ and $s \in S$ we have

$$\begin{aligned}
\mathbf{E}\left[\,\hat{g}_{s,i}(k)|\boldsymbol{x}(k)\right] &= \frac{N_s}{N_s-1}\mathbf{E}\left[\frac{C_s^+(k) - C_s^-(k) + \mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\middle|\boldsymbol{x}(k)\right] \\
&= \frac{N_s}{N_s-1}\mathbf{E}\left[\frac{\mathbf{E}\left[C_s^+(k) - C_s^-(k)|\boldsymbol{\Delta}(k)\right]}{\xi_s(k)\Delta_{s,i}(k)}\middle|\boldsymbol{x}(k)\right] \\
&= \frac{N_s}{N_s-1}\left(\mathbf{E}\left[\frac{\boldsymbol{\nabla}\Lambda_s^T(\boldsymbol{x}(k))\sum_{s'\in S} c_{s'}(k)\bar{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k)}{\xi_s(k)\Delta_{s,i}(k)}\middle|\boldsymbol{x}(k)\right]\right. \\
&\quad - \mathbf{E}\left[\frac{\boldsymbol{\nabla}\Lambda_s^T(\boldsymbol{x}(k))\sum_{s'\in S}\frac{c_{s'}(k)\sum_{j=1}^{N_{s'}}\Delta_{s,j}(k)}{N_{s'}}\boldsymbol{u}_{\boldsymbol{s'}}}{\xi_s(k)\Delta_{s,i}(k)}\middle|\boldsymbol{x}(k)\right] \\
&\quad \left. + \mathbf{E}\left[\sum_{s'\in S}\frac{O\left(c_{s'}^2(k)\boldsymbol{\Delta}_{\boldsymbol{s'}}^2(k)\right)}{c_s(k)\Delta_{s,i}(k)}\right]\right) \\
&= \frac{N_s}{N_s-1}\left(\frac{N_s-1}{N_s}\boldsymbol{\nabla}\Lambda_{s,i}(\boldsymbol{x}(k)) + O(\xi_s(k))\right) \\
&= \sum_{l\in L}\frac{\partial C_l(x^l(k))}{\partial x_{s,i}} + O(\xi_s(k)) \qquad (2.18) \\
&= \frac{\partial C(\boldsymbol{x}(k))}{\partial x_{s,i}} + O(\xi_s(k))
\end{aligned}$$

where $C_s^-(k) = \Lambda_s(\boldsymbol{x}(k))(= \sum_{l\in L^s} C_l(x^l(k)))$, $C_s^+(k) = \Lambda_s(\boldsymbol{x}(k) + \sum_{s'\in S} c_{s'}(k)\tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k))$,

$\boldsymbol{\nabla}\Lambda_s(\boldsymbol{x}(k)) := (\partial\Lambda_s(\boldsymbol{x}(k))/\partial x_{sp}; s \in S, p \in P_s)$, and

$$\tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) = \bar{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) - \frac{\sum_{j=1}^{N_{s'}} \Delta_{s',j}(k)}{N_{s'}} \boldsymbol{u}_{\boldsymbol{s'}} \ .$$

Therefore, one can see that $\boldsymbol{b(k)} \to 0$ with probability one.

From the assumption that $E[\mu_s^+(k) - \mu_s^-(k)|\mathcal{F}_k] = 0$ and using the independence of $\mu_s^{\pm}(k)$ and $\boldsymbol{\Delta_s}(k)$, we can bound the second moment of $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ as follows:

$$\begin{aligned} \mathbf{E}\left[(\hat{g}_{s,i}(k))^2\right] &= \mathbf{E}\left[\left(\frac{C_s^+(k) - C_s^-(k) + \mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2\right] \\ &= \mathbf{E}\left[\left(\frac{C_s^+(k) - C_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2 + \left(\frac{\mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2\right] \end{aligned} \tag{2.19}$$

Following a similar argument used above one can show that the first term in (2.19) is $O(1)$ and using the bounds on $\mathbf{E}\left[(\boldsymbol{\Delta_s}(k))^2\right]$, $\mathbf{E}\left[(\boldsymbol{\Delta_s}(k))^{-2}\right]$, and $\mathbf{E}\left[(\mu_s^{\pm}(k))^2\right]$ the second term is $O(\xi_s(k)^{-2})$. Therefore, $\sum_{i=k}^{\infty} \mathbf{E}\left[||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)||^2\right] < \infty$ w. p. 1. □

Since $\sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i)$, $j \geq k$ is a martingale and $\sum_{i=k}^{\infty} \mathbf{E}\left[||\boldsymbol{A}(i)\varrho(i)||^2\right] < \infty$, from Doob-Kolmogorov inequality ([17, p. 309]), for each $\epsilon > 0$, we have

$$P(\sup_{j \geq k} || \sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i)|| \geq \epsilon) \leq \frac{1}{\epsilon^2} \sum_{i=k}^{\infty} \mathbf{E}\left[|| \boldsymbol{A}(i)\boldsymbol{\varrho}(i)||^2\right] \ . \tag{2.20}$$

Because the right-hand side of (2.20) goes to zero as $k \to \infty$, it follows that

$$\lim_{k \to \infty} P(\sup_{j \geq k} || \sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i)|| \geq \epsilon) = 0 \ .$$

and the existence of the $\gamma(k)$ sequence is guaranteed.

We interpolate $\boldsymbol{x}(k)$ into a continuous parameter process $\boldsymbol{X^0}(t)$, which is used

to get the "mean" ODE, as follows:

$$\boldsymbol{X}^{\mathbf{0}}(t) := \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\boldsymbol{x}(k+1)\ ,\ t \in (t_k, t_{k+1}) \tag{2.21}$$

where $t_k = \sum_{i=0}^{k-1} \hat{a}(i)$.[6]

Let

$$\begin{aligned}
\boldsymbol{B}^{\mathbf{0}}(t_k) &:= \sum_{i=1}^{k-1} \boldsymbol{A}(i)\boldsymbol{b}(i), \quad & \boldsymbol{M}^{\mathbf{0}}(t_k) &:= \sum_{i=1}^{k-1} \boldsymbol{A}(i)\boldsymbol{\varrho}(i), \\
\mathbf{T}^{\mathbf{0}}(t_k) &:= \sum_{i=1}^{k-1} \boldsymbol{\tau}(i), & \boldsymbol{\Phi}^{\mathbf{0}}(t_k) &:= \sum_{i=1}^{k-1} \boldsymbol{\phi}(i),
\end{aligned}$$

and the corresponding piecewise linear interpolations are defined as

$$\begin{aligned}
\boldsymbol{B}^{\mathbf{0}}(t) &:= \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{B}^{\mathbf{0}}(t_k) + \frac{t-t_k}{\hat{a}(k)}\boldsymbol{B}^{\mathbf{0}}(t_{k+1}) \quad t \in (t_k, t_{k+1}), \\
\boldsymbol{M}^{\mathbf{0}}(t) &:= \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{M}^{\mathbf{0}}(t_k) + \frac{t-t_k}{\hat{a}(k)}\boldsymbol{M}^{\mathbf{0}}(t_{k+1}) \quad t \in (t_k, t_{k+1}), \\
\mathbf{T}^{\mathbf{0}}(t) &:= \frac{t_{k+1}-t}{\hat{a}(k)}\mathbf{T}^{\mathbf{0}}(t_k) + \frac{t-t_k}{\hat{a}(k)}\mathbf{T}^{\mathbf{0}}(t_{k+1}) \quad t \in (t_k, t_{k+1}), \\
\boldsymbol{\Phi}^{\mathbf{0}}(t) &:= \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{\Phi}^{\mathbf{0}}(t_k) + \frac{t-t_k}{\hat{a}(k)}\boldsymbol{\Phi}^{\mathbf{0}}(t_{k+1}) \quad t \in (t_k, t_{k+1}.)
\end{aligned}$$

[6]Recall that $\hat{a}(k) = \max_{s\in S} a_s(k)$.

Using (2.13), $\boldsymbol{X^0}(t)$ can be written as

$$\begin{aligned}
\boldsymbol{X^0}(t) &= \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\bigg(\boldsymbol{x}(k) + \boldsymbol{A}(k)\Big[-\boldsymbol{\nabla} C(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)\Big] \\
&\quad + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\bigg) \\
&= \boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\bigg(\boldsymbol{A}(k)\left[-\boldsymbol{\nabla} C(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)\right] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\bigg) \\
&= \boldsymbol{x}(1) + \sum_{i=1}^{k-1}\bigg(\boldsymbol{A}(i)[-\boldsymbol{\nabla} C(\boldsymbol{x}(i)) + \boldsymbol{b}(i) + \boldsymbol{\varrho}(i)] + \boldsymbol{\tau}(i) + \boldsymbol{\phi}(i)\bigg) \\
&\quad + \frac{t-t_k}{\hat{a}(k)}\bigg(\boldsymbol{A}(k)\left[-\boldsymbol{\nabla} C(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)\right] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\bigg) \\
&= \boldsymbol{x}(1) + \boldsymbol{B^0}(t) + \boldsymbol{M^0}(t) - \sum_{i=1}^{k-1}\hat{a}(i)\boldsymbol{\nabla} C(\boldsymbol{x}(i)) \\
&\quad - \frac{t-t_k}{\hat{a}(k)}\hat{a}(k)\boldsymbol{\nabla} C(\boldsymbol{x}(k)) + \boldsymbol{Z^0}(t) + \mathbf{T^0}(t) + \boldsymbol{\Phi^0}(t) \\
&= \boldsymbol{x}(1) + \boldsymbol{B^0}(t) + \boldsymbol{M^0}(t) + \boldsymbol{H^0}(t) + \boldsymbol{Z^0}(t) + \mathbf{T^0}(t) + \boldsymbol{\Phi^0}(t), \qquad (2.22)
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{Z^0}(t) &= \sum_{i=0}^{k-1}\big(\hat{a}(i) - \boldsymbol{A}(i)\big)\boldsymbol{\nabla} C(\boldsymbol{x}(i)) + \frac{t-t_k}{\hat{a}(k)}\big(\hat{a}(k) - \boldsymbol{A}(k)\big)\boldsymbol{\nabla} C(\boldsymbol{x}(k)), \\
\boldsymbol{H^0}(t) &= -\int_0^t \boldsymbol{\nabla} C(\bar{\boldsymbol{x}}(s))ds, \\
\bar{\boldsymbol{x}}(s) &= \boldsymbol{x}(k),\ s \in [t_k, t_{k+1})\ .
\end{aligned}$$

Since we are interested in the tail properties of the interpolated process, let us define the left shifted and centered process $\boldsymbol{X^k}(t)$:

$$\boldsymbol{X^k}(t) := \boldsymbol{x}(k) + \boldsymbol{B^k}(t) + \boldsymbol{M^k}(t) + \boldsymbol{H^k}(t) + \boldsymbol{Z^k}(t) + \mathbf{T^k}(t) + \boldsymbol{\Phi^k}(t), \qquad (2.23)$$

where

$$\begin{aligned}
&\boldsymbol{B^k}(t) = \boldsymbol{B^0}(t_k + t) - \boldsymbol{B^0}(t_k), \quad \boldsymbol{M^k}(t) = \boldsymbol{M^0}(t_k + t) - \boldsymbol{M^0}(t_k), \\
&\mathbf{T^k}(t) = \mathbf{T^0}(t_k + t) - \mathbf{T^0}(t_k), \qquad \boldsymbol{\Phi^k}(t) = \boldsymbol{\Phi^0}(t_k + t) - \boldsymbol{\Phi^0}(t_k), \\
&\boldsymbol{Z^k}(t) = \boldsymbol{Z^0}(t_k + t) - \boldsymbol{Z^0}(t_k), \qquad \boldsymbol{H^k}(t) = -\int_0^t \boldsymbol{\nabla} C(\bar{\boldsymbol{x}}(t_k + s)ds\ .
\end{aligned}$$

Similarly as in [28], we work with a fixed $w \notin \Omega_0$ where Ω_0 denotes the set for which $|| \boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \geq \gamma(k)/2$ infinitely often or the conditions given in Lemma 2.7.1 do not hold. In this case $\boldsymbol{M^k}(t)$ and $\boldsymbol{B^k}(t)$ converge to zero uniformly on finite intervals as $k \to \infty$. In addition, $\boldsymbol{\Phi^k}(t) \to 0$ uniformly on finite intervals as $k \to \infty$ because we have

$$||\boldsymbol{\phi}(k)|| \leq || - \boldsymbol{A}(k)\boldsymbol{\nabla}C(\boldsymbol{x}(k) + \boldsymbol{A}(k)\boldsymbol{\varrho}(k) + \boldsymbol{A}(k)\boldsymbol{b}(k)||(1 - I(k))$$
$$+||\Pi_\Theta[\boldsymbol{v}(k)] - \boldsymbol{x}(k)||(1 - I(k)).$$

and only a finite number of terms of $(1 - I(k))$ are nonzero.

Under assumption **A6**, using similar arguments one can easily verify that $\boldsymbol{Z^k}(t) \to 0$ as $k \to \infty$. Therefore, all the time varying terms on the right side of (2.23) other than $\boldsymbol{H^k}(t) + \mathbf{T^k}(t)$ go to zero as $k \to \infty$, and Assumption **A5** is sufficient to guarantee the equicontinuity of $\boldsymbol{H^k}(t)+\mathbf{T^k}(t)$ as shown in [28, Theorem 5.3.1]. Hence, as a consequence of Arzela-Ascoli Theorem ([27, p. 72]), there exists a convergent subsequence $\boldsymbol{X}(t) \to \mathbf{x}^*$, which in fact results in $\boldsymbol{x}(k) \to \mathbf{x}^*$, and as shown in [28, Theorem 5.3.1], $\mathbf{x}^*$ is determined by the solution of the following ODE:

$$\dot{\boldsymbol{x}} = \bar{\Pi}_\Theta[-\boldsymbol{\nabla}C(\boldsymbol{x})], \tag{2.24}$$

where

$$\bar{\Pi}_\Theta[-\boldsymbol{\nabla}C(\boldsymbol{x})] := \lim_{\delta \downarrow 0} \frac{\Pi_\Theta\big[\boldsymbol{x} - \delta\boldsymbol{\nabla}C(\boldsymbol{x})\big] - \boldsymbol{x}}{\delta}.$$

Note that the stationary points of the ODE above is the set of Kuhn-Tucker points

$$KT = \{\boldsymbol{x} : \exists\, \lambda^i \geq 0 \,\text{such that}\, \boldsymbol{\nabla}C(\boldsymbol{x}) + \sum_{i:\, \boldsymbol{q}_i(\boldsymbol{x})=0} \lambda^i \boldsymbol{q}_i(\boldsymbol{x}) = 0, \} \tag{2.25}$$

where $\boldsymbol{q}_i(\cdot)$ terms are the constraints of the problem. Since the cost function $C(\boldsymbol{x})$ is assumed to be convex, any KT point belongs also to $\mathbf{x}^*$ which concludes the proof.

2.8 Proof of Theorem 2.3.2

For the proof of the constant step size case, since the step size is fixed and the same for all SD pairs, we can directly apply the result of Theorem 8.2.1 of [28, p. 219]. Let $a_s(k) := a \leq 1$ and $\xi_s(k) := \xi = \frac{a \cdot \epsilon}{2\alpha}$. Then, as in the decreasing step size case with $k > K_1$, we can replace the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ with its linear approximation. Hence $\mathbf{E}\left[\hat{g}_{s,i}(k)|\boldsymbol{x}(k)\right] = \frac{\partial C(\boldsymbol{x}(k))}{\partial x_{si}} + O(\xi)$ and we have

$$\lim_{m,k,a} \frac{1}{m} \sum_{i=k}^{k+m-1} \mathbf{E}\left[O\left(\xi\right)|\mathcal{F}_k\right] = \lim_{m,k,a} \frac{1}{m} \sum_{i=k}^{k+m-1} \mathbf{E}\left[O\left(\frac{a.\epsilon}{2\alpha}\right)|\mathcal{F}_k\right] = 0 \text{ in mean.}$$

where the $\lim_{m,k,a}$ means that the limit is taken as $m \to \infty$, $k \to \infty$, and $a \to 0$ simultaneously in any way at all. Then, $\hat{\boldsymbol{g}}(k) = \{\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k), s \in S\}$ satisfies all the conditions given in Theorem 8.2.1 of [28, p. 219], establishing the convergence.

Chapter 3

A Unified Framework for Multipath Routing for Unicast and Multicast Traffic

3.1 Background & Set-up

Consider a network that consists of a set of unidirectional links $\mathcal{L} = \{1, \ldots, L\}$. Let $\mathcal{S} = \{1, \ldots, S\}$ be the set of source nodes. Each source node is associated with a session that can be either a unicast or a multicast session. For simplicity each source is assumed to have a single session. However, the results also apply to the case where there may be multiple sessions originating at a single source node. We use D^s to denote the set of destination nodes for the source $s \in \mathcal{S}$. Each source needs to deliver packets to every destination $d \in D^s$ at a fixed rate r^s. As discussed in Section 2.4.1, we study the scenario where the sources can make use of pre-installed application-layer overlay nodes that can be used to provide the sources with alternate paths in addition to the default path provided by the underlying IP routing protocol. We denote the set of *core* overlay nodes by $\mathcal{O}$ and the set of overlay nodes in $\mathcal{O}$ used to create *alternative* paths between a source $s \in \mathcal{S}$ and its destination node(s) in D^s by $O_c^s \subseteq \mathcal{O}$. Here a destination node refers to a unicast destination node or a multicast receiver node. Since every source node is also an (edge) overlay node, the set of overlay nodes utilized by a source $s \in \mathcal{S}$ is given by

$O^s := O_c^s \cup \{s\}$, and there are $N_s := |O^s|$ paths available to each destination node, where $|O^s|$ denotes the cardinality of O^s. Define $N = \sum_{s \in \mathcal{S}} (N_s \cdot |D^s|)$.

We are interested in designing a load balancing algorithm that can utilize multiple paths available between sources and destination nodes and optimize the network performance according to a given network cost function. In order to capture the performance improvement vs. cost trade-off with increasing network capabilities, we consider three different network models (described in Section 3.2). These models differ in the set of assumptions we impose on the capability of the underlying network. We study the relative performance of these systems and some of the properties of their operating points.

In the remainder of this section we explain how some of the bookkeeping issues that arise in our problem can be handled using a special family of codes, called Digital Fountain codes.

3.1.1 Digital Fountain codes

As discussed in Chapter 1, when a source s forwards packets to a destination d, without any special coding it must ensure that the destination receives all distinct packets necessary for recovering the message. When different sets of packets are forwarded to different destinations using two or more overlay nodes, the source must keep track of the packets forwarded along different paths so that every destination receives all necessary packets. As a result, this requires complicated bookkeeping at the multicast sources and the core overlay nodes. We propose the adoption of

Digital Fountain codes to solve this problem.

The original application area of Digital Fountain codes [31, 39] is the reliable transmission of data over the Internet as an alternative to the unicast TCP/IP retransmissions. Since the Internet can be modeled as an erasure channel, the idea is to use an erasure-correcting code that will eliminate retransmissions. Classic block codes for erasure correction are called Reed-Solomon codes. An $(N;K)$ Reed-Solomon code has the ideal property that if any K of the N transmitted symbols are received then the original K source symbols can be recovered. However, when using a Reed-Solomon code, as with any block code, one must estimate the erasure probability and choose the code rate $R = K/N$ before transmission. In addition, Reed-Solomon codes are practical only for small K, N.

Fountain codes are rateless in the sense that the number of encoded packets that can be generated from the source message is potentially limitless; the number of encoded packets to be generated can be determined on the fly. Regardless of the statistics of the erasure events on the channel, one can send as many encoded packets as needed in order for the decoder to recover the source data. The input and output symbols can be bits or more generally binary vectors of arbitrary length. Each output symbol is generated by a (binary) addition of some randomly selected input symbols. The number of input symbols to be added is determined according to some fixed degree distribution. It is assumed that each output symbol is tagged with information describing which input symbols are used to generate it, for example, in the packet header.

A decoding algorithm for a Fountain code is an algorithm that recovers the

original K input symbols from *any* set of M output symbols with a high probability. For good Fountain codes the value of M is very close to K and the decoding time is approximately linear in K. Raptor codes [39] are examples of such Digital Fountain codes with linear time encoders and decoders for which the probability of decoding failure converges to zero polynomially fast in the number of input symbols. For instance, for $K = 64{,}536$ and $M = 65{,}552$, i.e., with a redundancy of 1.5 percent, it is shown in [39] that the error probability is upper bounded by 1.71×10^{-14}. In practice, however, Digital Fountain codes introduce approximately 5 percent overheads.

Raptor codes have been used in commercial systems by the Digital Fountain startup company. Their Raptor code implementation can encode packets at the speed of several gigabits per second on a 2.4 Ghz Intel Xeon processor with a fairly small upper bound on the decoding error probability even for a small number of input symbols.

In our formulation we assume that a source first divides the traffic into blocks of, say, K symbols and applies a form of Digital Fountain code (e.g., Raptor code) to generate encoded output symbols that are forwarded to the destinations. Here the block size will be constrained by the buffer size at the source. Since a receiver can recover the K source symbols in each block from any M encoded symbols, the source node does not require any bookkeeping as long as it sends *distinct* packets along each path. This will guarantee that each receiver successfully receives the whole multicast stream as long as each user receives packets at a sufficient rate. This allows us to formulate our problem as one of assigning packet forwarding rates to available paths

for each destination, subject to a constraint that the aggregate rate at which the destination receives packets exceeds certain threshold. This threshold depends on the demand rate r^s as well as the efficiency of the coding scheme.

3.2 Network models

In this section we describe three different models that we consider for performance evaluation. For each $s \in \mathcal{S}$ and $d \in D^s$ let $x^s_{o,d}$ be the rate at which the source node s sends packets to destination d through an overlay node $o \in O^s$.[1] Also, define x^s_o to be the total rate at which an overlay node o receives packets from source s. In a unicast case this is simply the rate at which packets are forwarded to the destination through the overlay node, while in the case of a multicast session, it depends on the capability of the underlying network and the implementation as will be explained shortly.

As mentioned in the previous subsection, the adoption of a Digital Fountain code reduces our problem to that of rate assignment $\boldsymbol{x} = (x^s_{o,d}, s \in \mathcal{S}, o \in O^s, d \in D^s)$, which is the focus of the remainder of this chapter. We assume that the overlay nodes can copy packets. Hence, the sources need to deliver only a single copy of the packets to an overlay node, and the overlay node acts as a surrogate source for these packets. Under this assumption, the rate x^s_o to an overlay node o is given by $x^s_o = \max_{d \in D^s} x^s_{o,d}$, and depending on the assumed network model and the assigned

[1] When a source s uses a default path or a multicast tree rooted at itself to deliver some of packets to the receivers, we say that the source uses itself as an overlay node to forward the packets, although strictly speaking no processing by any overlay node is involved for these packets.

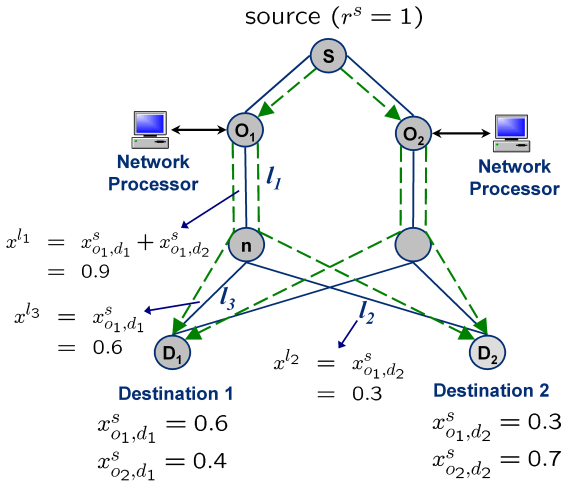


Figure 3.1: Network model-I (NM-I). Each dotted line represents a unicast session.

rates, some or all of the packets forwarded to the overlay node are relayed to the destinations.

We now describe the models in increasing order of network capabilities.

3.2.1 Network Model-I

The first network model we consider represents a unicast only IP network, where routers do not possess IP multicast functionality. We assume that packets are encoded using a Digital Fountain code at the source. A source node first forwards the encoded packets to overlay nodes at the required rate, and the overlay nodes create a unicast session for *each* destination and forward packets at the specified rate $x^s_{o,d}$. Hence, a source node and overlay nodes need to maintain multiple unicast sessions in case of a multicast session with more than one destination. This is shown in Figure 3.1.

Let $V^{n_1}_{n_2} \subset \mathcal{L}$ be the set of links in the default path from node n_1 to node n_2.

Given a rate assignment $\boldsymbol{x}$, the link loads $x^l, l \in \mathcal{L}$, under this network model are given by

$$x^l = \sum_{s \in S} \left(\sum_{o \in O_c^s : l \in V_o^s} x_o^s + \sum_{o \in O^s} \Big(\sum_{d \in D^s : l \in V_d^o} x_{o,d}^s \Big) \right) \tag{3.1}$$

We refer to this model as NM-I.

3.2.2 Network Model-II

Under Network Model-II the routers are IP multicast capable. We assume that each overlay node $o \in O^s$ creates a separate multicast tree $\mathcal{MT}_o^s$ rooted at itself for forwarding packets from the source s, using an intradomain multicast algorithm (e.g., DVMRP [46]). However, we assume that IP multicast routers are only capable of copying and forwarding packets. Hence, every packet forwarded to an overlay node by a source node s is relayed to *all* destinations in D^s. As a result, the rate at which destination nodes receive packets from an overlay node is the same (assuming no packet losses) and is given by $x_o^s = \max_{d \in D^s} x_{o,d}^s$. This is shown in Figure 3.2. Clearly, this may cause a receiver to receive packets at a rate larger than the intended rate. However, as we will show shortly, our algorithm exploits this property through measurements and attempts to eliminate such redundancy. In fact, at the optimal operating point $\boldsymbol{x}^\star$ we have ${x_{o,d}^s}^\star = x_o^{s\star}$ for all $d \in D^s$.

Under this model the load of link l can be written as

$$x^l = \sum_{s \in \mathcal{S}} \left(\sum_{o \in O_c^s : l \in V_o^s} x_o^s + \sum_{o \in O^s : l \in T_o^s} x_o^s \right) \tag{3.2}$$

where T_o^s is the set of links in the multicast tree $\mathcal{MT}_o^s$ in the case of a multicast session or the set of links in the default path in the case of a unicast session, *i.e.,* $T_o^s =$

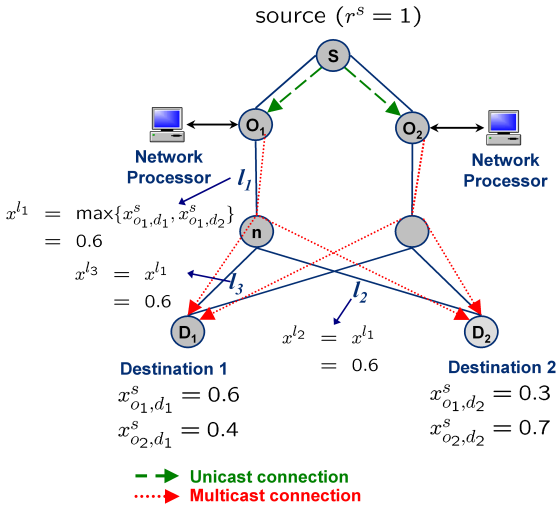


Figure 3.2: Network model-II (NM-II).

V^o_d. This model is referred to as NM-II.

3.2.3 Network Model-III

In this model, in addition to the IP multicast capability we also assume that each router is capable of forwarding packets onto each branch at a different rate. We refer to these routers as "smart" routers to distinguish them from the routers used in NM-II. This is shown in Figure 3.3. Under this model a source s can select the individual rates $x^s_{o,d}$ independently for each destination, and packets will be forwarded to a destination $d \in D^s$ at the intended rate $x^s_{o,d}$ (as opposed to $\max_{d' \in D^s} x^s_{o,d'}$ under NM-II).[2] This allows the network operator more flexibility and fine-grained control in rate assignment and to better exploit the existence of multiple paths through overlay nodes, while making use of multicast nature of the traffic at

[2]This assumes that there are no packet losses along the path.

the same time.

The link rates under this model are given by

$$x^l = \sum_{s \in S} \left(\sum_{o \in O_c^s : l \in V_o^s} x_o^s + \sum_{o \in O^s} \max_{d \in D^s : l \in \hat{V}_d^o} x_{o,d}^s \right) . \tag{3.3}$$

Here $\hat{V}_d^o$ denotes the set of links along the path from overlay node o to destination d. In the case of a multicast session, this is the set of links in the multicast tree, which may be different from the default path provided by the underlying routing protocol. We will refer to this model as NM-III.

Due to the large gap between the level of intelligence currently available at the IP routers and the required intelligence assumed in this model, it is unlikely that this type of network will be available in the near future. However, we consider this model for comparison purposes, and compare the performance of the other models against that of this somewhat ideal model.

Note that under these models, overlay nodes can be viewed as content delivery servers that store a portion of the original content to be distributed. Our goal is to design a *unified* load balancing algorithm that minimizes the total network cost by distributing the traffic load among multiple available paths, under *all* three network models. However, since the link loads depend on the assumed network model, the desired operating point as well as the aggregate network cost are also determined by the assumed network model. This will allow us to quantify the additional net benefit we can acquire from placing increasing network capabilities and to carry out the performance vs. cost trade-off analysis.

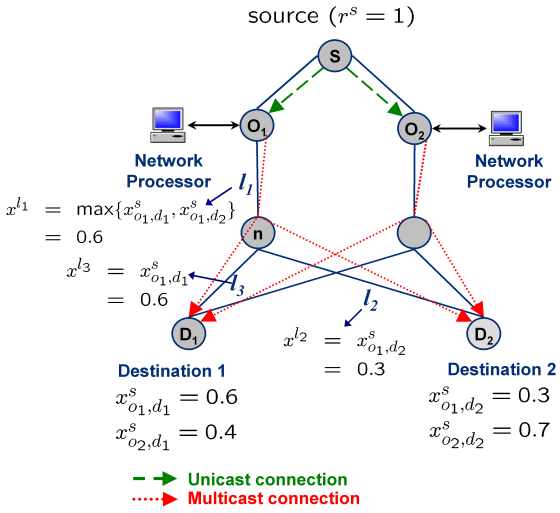


Figure 3.3: Network model-III (NM-III).

3.3 Optimization Framework and the Proposed Algorithm

We formulate the problem of rate assignment as an optimization problem, where the objective function is the sum of link costs. A link cost is a function of the total rate traversing the link x^l and is given by $C_l(x^l), l \in \mathcal{L}$. This formulation is similar to the one defined in Chapter 2 for the unicast traffic. As before the link cost functions are assumed to be convex. However, here we need to drop the differentiability assumption due to the fact that the link rate functions (3.1), (3.2) and (3.3) are not differentiable with respect to the input traffic rates $x^s_{o,d}$. The optimization problem can be stated as follows:

$$\min_{\boldsymbol{x}} C(\boldsymbol{x}) = \sum_{l \in \mathcal{L}} C_l(x^l) \tag{3.4}$$

$$\text{s.t. } \sum_{o \in O_s} x^s_{o,d} = r^s + \epsilon^s, \forall s \in \mathcal{S}, d \in D^s \tag{3.5}$$

$$x^s_{o,d} \geq \nu, \ \forall s \in \mathcal{S}, o \in O^s, d \in D^s \tag{3.6}$$

where r^s is the assumed traffic rate of source s, ν is an arbitrarily small positive constant,[3] and ϵ^s is the additional rate required by the coding scheme for a receiver to successfully decode the incoming encoded data.

This problem in (3.4) can be viewed as a natural generalization of the problem studied in Chapter 2. Indeed, the algorithm proposed in Chapter 2 is a special case of the algorithm described in this paper with only unicast sessions under NM-I described in subsection 3.2.1. There are two major differences between the current problem and that investigated in the previous chapter. First, here the link loads $x^l, l \in \mathcal{L}$, resulting from the rate assignment, and hence the overall network cost, depend on the adopted network model. Consequently, the performance of the system depends on the assumed capability of the underlying network. Second, the lack of differentiability of the cost function with respect to the rate assignment $\boldsymbol{x}$ raises some technical issues as will be clear in the analysis.

3.3.1 Proposed routing algorithm

One thing to note in the optimization problem in (3.4) - (3.6) is that the decision variable $\boldsymbol{x}$ is a collection of rate assignments of the sources $x^s, s \in \mathcal{S}$, and the constraints given in (3.5) and (3.6) comprise separate constraints for each source that are independent of others. Therefore, the problem can be naturally decomposed into several coupled subproblems, one for each source.

Let Θ_s denote the set of feasible rate assignments for source s that satisfy the

[3]For instance, some of the control packets may be routed along different paths available between the source and destination nodes.

constraints in (3.5) - (3.6) and $\Pi_{\Theta_s}[\boldsymbol{\zeta}]$ the projection of a vector $\boldsymbol{\zeta}$ onto the feasible set Θ_s using the Euclidean norm. We denote the set of links utilized by source s's packets by L^s. Clearly, this set L^s depends on the assumed network model and is given by $\{V_o^s \cup V_d^o : o \in O^s, d \in D^s\}$ for NM-I and $\{V_o^s \cup T_o^s : o \in O^s\}$ for NM-II and NM-III.

In order to find a solution to (3.4) we propose the following SPSA-based algorithm to be run at each source node in a distributed manner: At time $k = 0, 1, \ldots$, each source s updates its rate vector $\boldsymbol{x_s}(k)$ according to

$$\boldsymbol{x_s}(k+1) = \Pi_{\Theta_s}[\boldsymbol{x_s}(k) - a_s(k)\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)] \tag{3.7}$$

where $a_s(k) > 0$ is the step size, and $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ is an approximation to either the gradient vector $\boldsymbol{\nabla} C_s(k) = (\partial C(\boldsymbol{x}(k))/\partial x_{o,d}^s, o \in O^s, d \in D^s)$ if $C(x(k))$ is differentiable or a subgradient vector $sg(x)$ otherwise [20].

In our proposed algorithm we compute $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ according to

$$\begin{aligned} \hat{g}_{s,i}(k) : &= \frac{N_s}{N_s - 1} \frac{y_s(\Pi_\Theta[\boldsymbol{x}(k) + \Xi(k)\boldsymbol{\Delta}(k)]) - y_s(\boldsymbol{x}(k))}{\xi_s(k)\Delta_{s,i}(k)} \\ &= \frac{N_s}{N_s - 1} \frac{(C_s^+(k) + \mu_s^+(k)) - (C_s^-(k) + \mu_s^-(k))}{\xi_s(k)\Delta_{s,i}(k)}, \\ & \qquad i = 1, \ldots, N_s \cdot |D^s|, \end{aligned} \tag{3.8}$$

Similar to the discussion in Chapter 2, $\boldsymbol{\Delta}(k) = (\boldsymbol{\Delta_s}(k),\ s \in S)$ is an $N \times 1$ vector, $\boldsymbol{\Delta_s}(k)$ is the random perturbation vector generated by source s at iteration k, $\Xi(k)$ is an $N \times N$ diagonal matrix composed of block diagonal entries $\{\Xi_s(k) := \xi_s(k) \cdot I_s, s \in \mathcal{S}\}$ with $\xi_s(k) > 0$ and I_s being the $(N_s \cdot |D^s|) \times (N_s \cdot |D^s|)$ identity matrix. We denote by $y_s(\boldsymbol{x})$ the noisy measurements of the *partial* network

cost $\Lambda_s(\boldsymbol{x}) := \sum_{l \in L^s} C_l(x^l)$ obtained with a given rate assignment vector $\boldsymbol{x}$. The variables $C_s^-(k)$ and $C_s^+(k)$ denote $\Lambda_s(\boldsymbol{x}(k))$ and $\Lambda_s(\Pi_\Theta[\boldsymbol{x}(k) + \Xi(k)\boldsymbol{\Delta}(k)])$, respectively, and $\mu_s^+(k)$ and $\mu_s^-(k)$ represent the measurement noises due to stochastic nature of traffic and/or potential lack of synchronization of the algorithms at the sources and will be modeled as random variables (rvs). For the simplicity of analysis, we assume that the distribution of the noise $\frac{\mu_s^+(k) - \mu_s^-(k)}{\Delta_{s,i}(k)}$ in the approximation is conditionally independent of the perturbation vector $\boldsymbol{\Delta}_s(k)$ selected by the source throughout this paper. In addition, we assume that a source keeps drawing a new perturbation vector until $\Pi_{\Theta_s}[\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)] \neq \boldsymbol{x_s}(k)$.

Note that this algorithm holds the properties that are discussed on Section 2.3.1 as well.

3.4 Convergence of the Proposed Algorithm

In this section we establish the convergence of the proposed algorithm in (3.7) and (3.8). We first consider the case where the step sizes $\{a_s(k), k = 1, 2, \ldots\}$ are decreasing and establish the a.s. convergence of the algorithm to a solution of the optimization problem in (3.4) - (3.6). Then, we study the case of a fixed or constant step size, i.e., $a_s(k) = a$ for all $s \in \mathcal{S}$ and $k = 0, 1, \ldots$, and demonstrate the weak convergence of the algorithm to a neighborhood of a solution.

3.4.1 Decreasing step size case

In this subsection we establish the a.s. convergence (or convergence w. p. 1) of (3.7) under all three network models described in Section 3.2.

The following definition is borrowed from convex analysis [20], and further details can be found in [35].

Definition 3.4.1. *Suppose that* $\boldsymbol{h}$ *is a real-valued convex function on* $\mathbb{R}^r$. *A vector* $\boldsymbol{sg}(\boldsymbol{x})$ *is said to be a subgradient of* $\boldsymbol{h}$ *at a point* $\boldsymbol{x}$ *if* $\boldsymbol{h}(\boldsymbol{z}) \geq \boldsymbol{h}(\boldsymbol{x}) + (\boldsymbol{z} - \boldsymbol{x})^T \boldsymbol{sg}(\boldsymbol{x})$ *for all* $\boldsymbol{z} \in \mathbb{R}^r$. *The set of all subgradients of* $\boldsymbol{h}$ *at* $\boldsymbol{x}$ *is called the subdifferential of* $\boldsymbol{h}$ *at* $\boldsymbol{x}$ *and is denoted by* $\partial \boldsymbol{h}(\boldsymbol{x})$.

In order to establish the convergence of our algorithm based on SPSA, we replace assumptions **A1** and **A2** (Section 2.3.1) with the following:

A6. The link cost functions $C_l\left(x^l\right)$ are convex for all $l \in L$, but are not necessarily differentiable. The subdifferential of C at $\boldsymbol{x}$ is denoted by $\partial C(\boldsymbol{x})$ and is bounded for all $\boldsymbol{x} \in \Theta$, where Θ is the feasible set of $\boldsymbol{x}$, i.e., $\boldsymbol{x_s} \in \Theta_s$ for all $s \in \mathcal{S}$.

A7. The perturbation terms $\Delta_{s,i}(k)$ are (i) mutually independent with zero mean for all $s \in S$ and $i \in \{1, 2, \cdots, N_s \cdot |D^s|\}$, (ii) uniformly bounded by some constant $\alpha < \infty$ with support on finite discrete sets, (iii) independent of $(\boldsymbol{x}(n), n = 0, 1, \cdots, k)$, and (iv) $\mathbf{E}\left[(\Delta_{s,i}(k))^{-1}\right]$, $\mathbf{E}\left[(\Delta_{s,i}(k))^{-2}\right]$ are bounded for all k.

Note that **A6** drops the differentiability assumption, while **A7** adapts **A2** to a multicast traffic problem by replacing source destination path variables (*i.e.,* P_s) with paths between a multicast source and it's destinations (*i.e.,* $\{1, 2, \cdots, N_s \cdot |D^s|\}$). The main result of this subsection is given by the following theorem.

Theorem 3.4.2. *Under Assumptions* ***A3*** *-* ***A7****, the sequence* $\boldsymbol{x}(k) = (\boldsymbol{x_s}(k),\ s \in S)$ *generated by the algorithm given by* (3.7) *converges to a point in the set of solutions of* (3.4) - (3.6) *w. p. 1 under each of the three network models with link loads defined by* (3.1) - (3.3)*, starting from any initial rate assignment* $(\boldsymbol{x_s}(0), s \in \mathcal{S}) \in \Theta$*.*

Proof: We provide a proof only under NM-I with link loads given by (3.1) in Section 3.7. The proof for the other two models follows from the fact that the necessary convexity of the objective function can be established in a similar manner. □

One thing to note here is that the proposed algorithm does not require any modifications for its convergence under different network models. This allows us to compare different network models using the same optimal routing algorithm and quantify the benefits obtained from increasing multicast capabilities. For the same reason the underlying IP network can be upgraded gradually without requiring any changes to our algorithm.

A property of NM-II

As mentioned in subsection 3.2.2, under NM-II all destinations receive packets at the same rate $\max_{d \in D^s} x^s_{o,d} = x^s_o$ from overlay node $o \in O^s$ because the multicast routers are assumed to be capable only of copying and forwarding packets. Based

on this observation one can trivially prove the following corollary.

Corollary 3.4.3. *Let $\boldsymbol{x}^\star$ be a feasible solution of* (3.4) *to which the a.s. convergence in Theorem 3.4.2 takes place under NM-II with link loads defined by* (3.2)*. Then, for all $s \in \mathcal{S}$ and $o \in O^s$, we have*

$$x_{o,d}^{s\star} = x_o^{s\star} \quad \textit{for all} \quad d \in D^s.$$

This simple result tells us that under NM-II the optimization problem in (3.4) - (3.6) can be reduced to that of finding the optimal rate assignments $(x_o^{s\star}, s \in \mathcal{S}, o \in O^s)$ to the overlay nodes. This is because the rates to individual receivers from an overlay node are the same as the rate from the source node to the overlay node. Therefore, the optimization problem in (3.4) - (3.6) can be rewritten as the following simpler optimization problem under NM-II.

$$\min_{\boldsymbol{x}} C(\boldsymbol{x}) = \min_{\boldsymbol{x}} \sum_{l \in \mathcal{L}} C_l(x^l) \tag{3.9}$$

$$\text{s. t.} \sum_{o \in O_s} x_o^s = r^s, \ \forall \ s \in S \tag{3.10}$$

where, with a little abuse of notation, $\boldsymbol{x} = (x_o^s, s \in \mathcal{S}, o \in O^s)$. When the number of receivers is large, this leads to much lower computational requirement at the sources and faster convergence as will be demonstrated in Section 3.5.

Note that in (3.10) the term ϵ^s is removed from (3.5). This is due to the fact that, at a feasible solution, a source node can deliver packets to the overlay nodes that simply forward the packets to all destinations. As a result, there is no issue of bookkeeping at the source and ϵ^s can be set to zero. We refer to this variation of NM-II as NM-IIb in the rest of the paper.

3.4.2 Constant step size case

Similar to the discussion in Section 2.3.2, we are interested in using a constant step size due to its practical implications.

We consider the case where the step sizes at the sources are fixed at $a_s(k) = a > 0$ and $\xi_s(k) = \xi$ for all $s \in \mathcal{S}$ and $k = 1, 2, \ldots$. Throughout this subsection we assume that $\xi := \frac{a \cdot \nu}{2\alpha}$, where α is the uniform bound on the perturbation terms $\Delta_{s,i}$ in Assumption **A2**. Although we are not able to establish the same a.s. convergence to a solution with constant step sizes, as shown in [27] under certain conditions constant step size SA algorithms can achieve weak convergence to a neighborhood of the solution set. Since the performance near the set of solutions is comparable to that of a solution in our problem, a constant step size policy is expected to perform reasonably well, which is supported by our simulation results in Section 3.5.

In this subsection we are interested in the system behavior when the step size is sufficiently small. To this end, we consider the following parametric scenario: Let $\{a_n, n = 1, 2, \ldots\}$ and $\{q_n, n = 1, 2, \ldots\}$ be a decreasing sequence of positive real numbers with $a_n \to 0$ and a nondecreasing sequence of non-negative integers, respectively.

Fix $n = 1, 2, \ldots$. We rewrite the update rule in (3.7) for all sources in the following compact form:

$$\begin{aligned} \boldsymbol{x}^n(k+1) &= \Pi_\Theta[\boldsymbol{x}^n(k) - a_n \hat{\boldsymbol{g}}^{\boldsymbol{n}}(k)] \\ &= \boldsymbol{x}^n(k) - a_n \hat{\boldsymbol{g}}^{\boldsymbol{n}}(k) + \boldsymbol{\upsilon}^n(k) \ , \end{aligned}$$

where $\boldsymbol{\upsilon}^n(k)$ is the reflection term to keep $\boldsymbol{x}^n(k+1)$ in the feasible set Θ.[4]

Let us define the following piece-wise constant continuous-time processes:

$$\boldsymbol{X}^n(t) \;=\; \begin{cases} a_n \sum_{i=q_n}^{\lfloor t/a_n \rfloor + q_n - 1} \boldsymbol{x}^n(i) \;, & \text{for } t \geq 0 \\ -a_n \sum_{i=\lfloor t/a_n \rfloor + q_n}^{q_n - 1} \boldsymbol{x}^n(i) \;, & \text{for } t < 0 \end{cases}$$

The process $\boldsymbol{\Upsilon}^n(t)$ is defined similarly with $\boldsymbol{\upsilon}^n(k)$ in place of $\boldsymbol{x}^n(k)$.

Before we state the main result of this subsection we impose the following additional assumption on $\hat{\boldsymbol{g}}(k), k = 0, 1, \ldots$.

A8. $\{\hat{\boldsymbol{g}}^n(k); n, k\}$ is uniformly integrable.

The main result of this subsection is now given by the following theorem.

Theorem 3.4.4. *Suppose that Assumptions **A3** and **A6**-**A8** hold. Then, for each subsequence of* $\{\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot), \quad n \in \mathbb{N} := \{1, 2, \ldots\}\}$, *there exists a further subsequence* $\{\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot), \quad n \in K\}$ *and a process* $(\boldsymbol{X}(\cdot), \boldsymbol{\Upsilon}(\cdot))$ *such that the following distributional convergence (denoted by* $\Rightarrow_n$*) takes place:*

$$(\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot)) \Rightarrow_n (\boldsymbol{X}(\cdot), \boldsymbol{\Upsilon}(\cdot)) \;,$$

where

$$\dot{\boldsymbol{X}} \in \partial C(\boldsymbol{X}) + \boldsymbol{\Upsilon}, \quad \boldsymbol{\Upsilon} \in -V(\boldsymbol{X}(t))$$

for almost all ω *(in the sample space* Ω *of the underlying probability space) and* t. *Here* $V(\boldsymbol{x})$ *denotes the convex cone generated by the set of outward normals defined in Section 3.7.*

[4]We use a superscript n to denote the dependency on the step size a_n.

Furthermore, for any $\delta > 0$, the fraction of time that $\boldsymbol{X}^n(a_n q_n + \cdot)$ spends in $N_\delta(S_\Theta)$ on [0, T] goes to one (in probability) as $n \to 0$ and $T \to \infty$, where S_Θ is the set of stationary points defined in Section 3.7 and $N_\delta(S_\Theta) := \{\boldsymbol{x} \in \Theta : \inf_{\boldsymbol{x}^\star \in S_\Theta} ||\boldsymbol{x} - \boldsymbol{x}^\star|| \leq \delta\}$.

Proof: The proof of Theorem 3.4.4 is given in Section 3.8. □

Theorem 3.4.4 states that for sufficiently small step sizes a, as $k \to \infty$, the rate assignment $\boldsymbol{x}(k)$ oscillates around a small neighborhood of the set of optimal solutions.

3.5 Simulation Results

In this section we evaluate the performance of our proposed algorithm under various network conditions and validate its convergence. In addition, we also compare the performance of our algorithm under the three network models to quantify the benefits with increasing level of intelligence at the routers, as well as that of DVMRP for comparison.

We developed a packet level discrete-event simulator for our study. For the optimization problem, the link cost function is selected to be $\left(x^l/c^l\right)^2$, where c^l is the link capacity and x^l is the link rate as defined in Section 3.2. In all simulations, the measurement or sampling period is set to one second. Therefore, source nodes can update their rates at most approximately every two seconds since two measurements are needed for estimating the (sub)gradient vector according to (3.8). For simplicity we set the source coding overhead/redundancy ϵ^s to zero. Experiments

are conducted with two different intradomain network topologies.

3.5.1 First topology

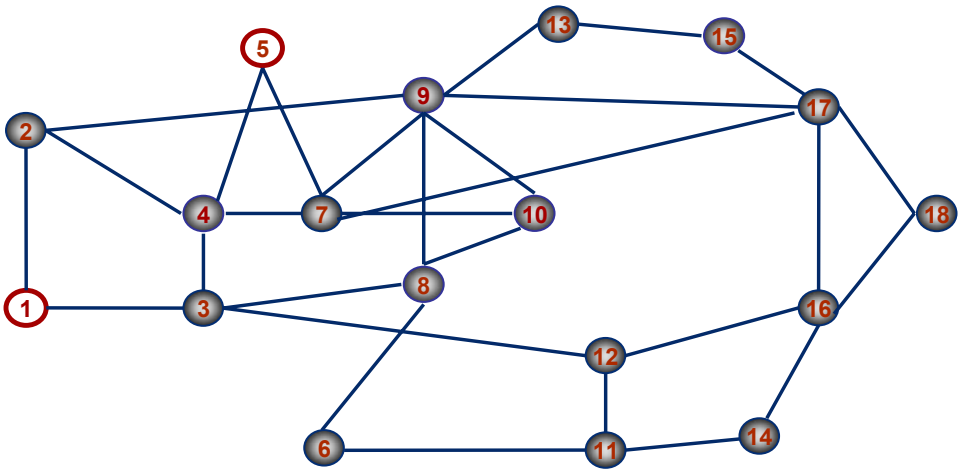


Figure 3.4: Network topology 1

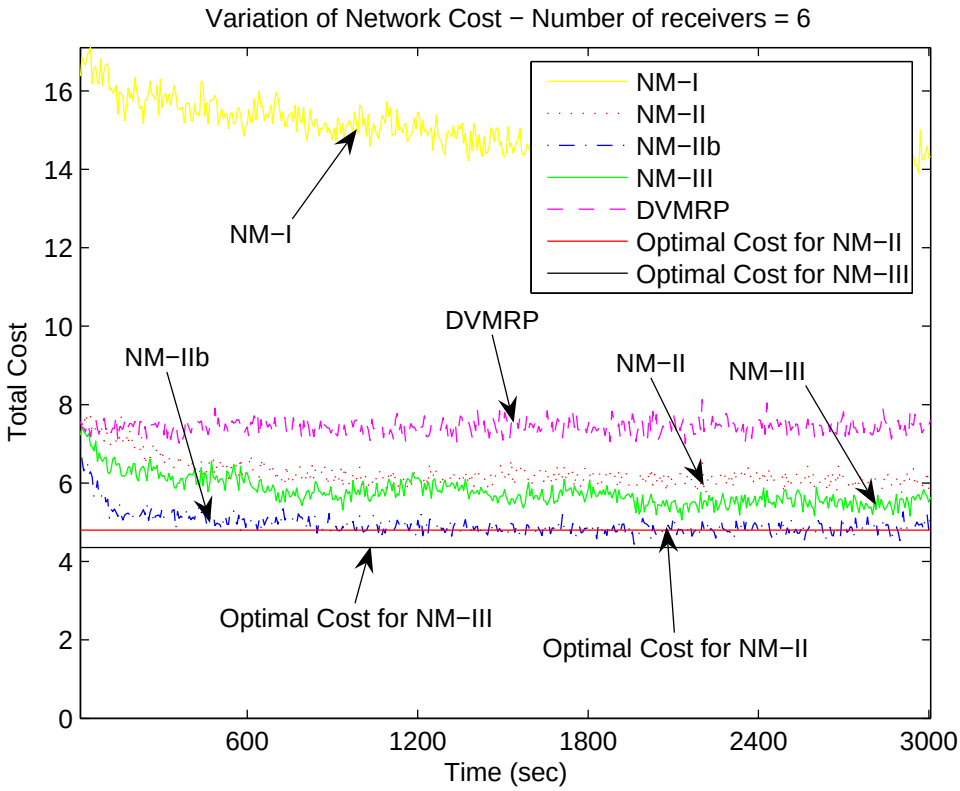
The first topology is shown in Figure 3.4. This topology is also used in [3, 32] and closely resembles the MCI backbone topology reported in [30]. Each link has a capacity of 20 Mbps. Packet size is selected to be 500 bytes. Nodes 1 and 5 are selected as multicast sources. In the first set of simulations, each source has 6 receivers. For each source we select a set of receivers spatially distributed throughout the network. In this example, $D^1 = \{4, 7, 8, 11, 13, 16\}$ and $D^5 = \{1, 8, 9, 12, 15, 17\}$. Nodes 9 and 17 are selected to be core overlay nodes and, hence, there are three available paths to reach each destination. For the simulations we attempt to select nodes with a higher degree as core overlay nodes. Each source generates Poisson traffic with an average rate of 11.5 Mbps.[5] Under all network models the rates along

[5]Since we focus on intra-domain routing, this rate may represent an aggregate rate of multiple multicast sources having the same receiver set D^s.

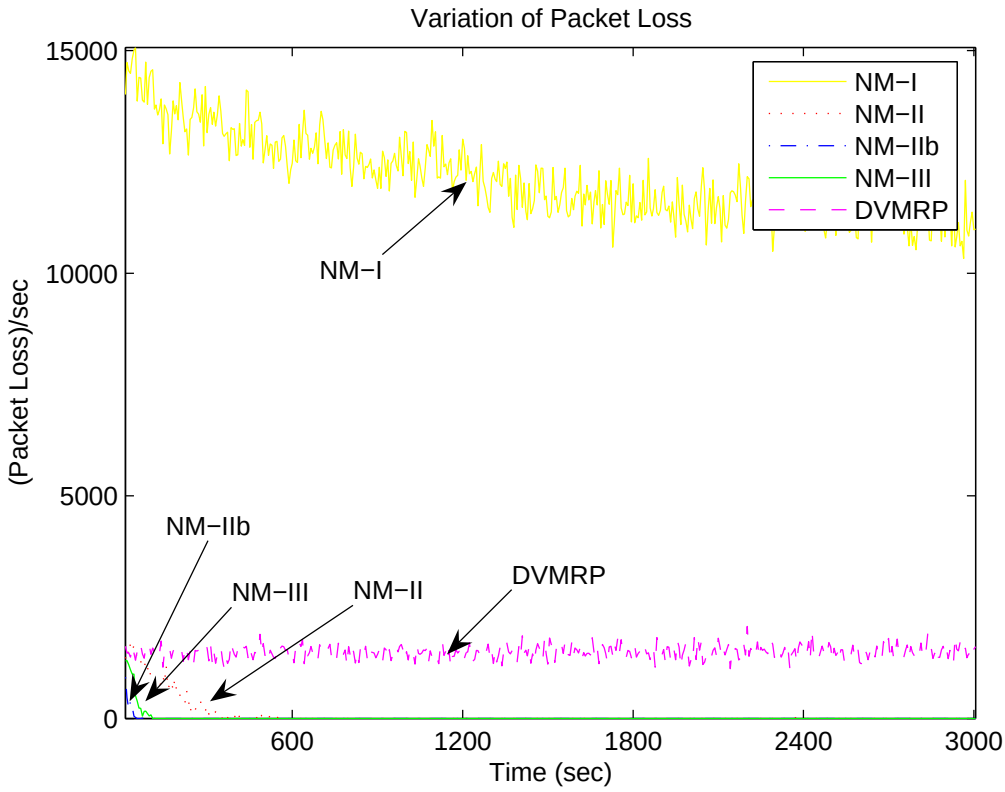
alternate routes through core overlay nodes are initially set to zero, i.e., $x^s_{o,d}(0) = 0$ if $o \neq s$ and $x^s_{s,d}(0) = r^s$. Hence, under NM-I model, the algorithm starts with traditional unicast routing to each destination, while under NM-II, NM-IIb and NM-III all the traffic is initially routed through the multicast tree rooted at the source and is gradually shifted to alternative trees rooted at core overlay nodes 9 and 17 when desired. In this simulation a decreasing step size policy is adopted, which satisfies the conditions in Assumptions **A3**-**A5**. In particular, $a_s(k) = 20/(k+100)^{0.602}$ and $\xi_s(k) = 50/(k^{0.101})$.

Figure 3.5 shows the evolution of total network cost and packet loss rate under different network models. Throughout this section we only plot a single sample path out of ten independent runs starting with different random seeds. The other sample paths are similar. We also compute the optimal cost values under NM-II and NM-III, using a MATLAB optimization package. The optimal values under NM-II and NM-III are 4.7979 and 4.3548, respectively. As expected the optimal value under NM-III is smaller than under NM-II due to assumed additional capabilities of the routers. However, interestingly the difference is rather small in this example. Hence, the additional complexity of having smart routers capable of forwarding packets onto each branch at a different rate offers only a marginal benefit in this case.[6] Also,

[6]This is not to say that the same holds in all cases as the difference depends on the topology and source-destination pair selections as well as their traffic demand. Indeed, we suspect that when there is a severe bottleneck along a path from an overlay node to a destination that can be congested under NM-II, then the optimal cost under NM-II can be significantly larger than that under NM-III.



(a)



(b)

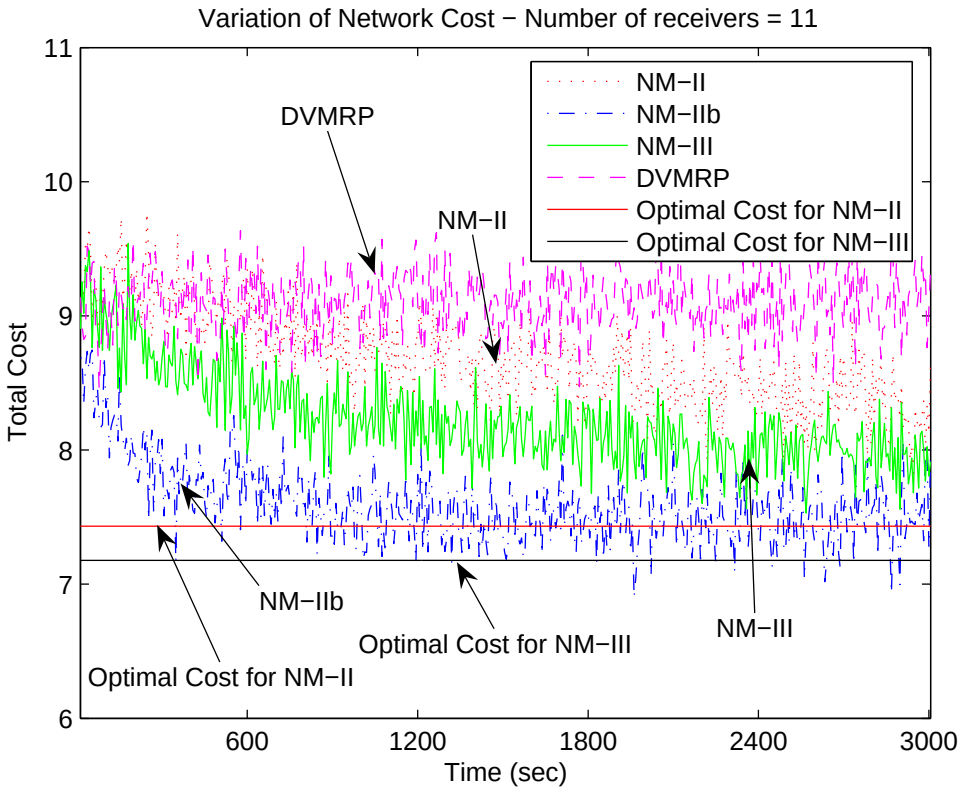
Figure 3.5: Network costs and packet losses with 6 receivers. (a) Network cost, (b) packet losses.

it is clear from the achieved network cost and packet loss rate that our algorithm considerably outperforms DVMRP under both NM-II and NM-III by distributing the traffic among three available multicast trees.

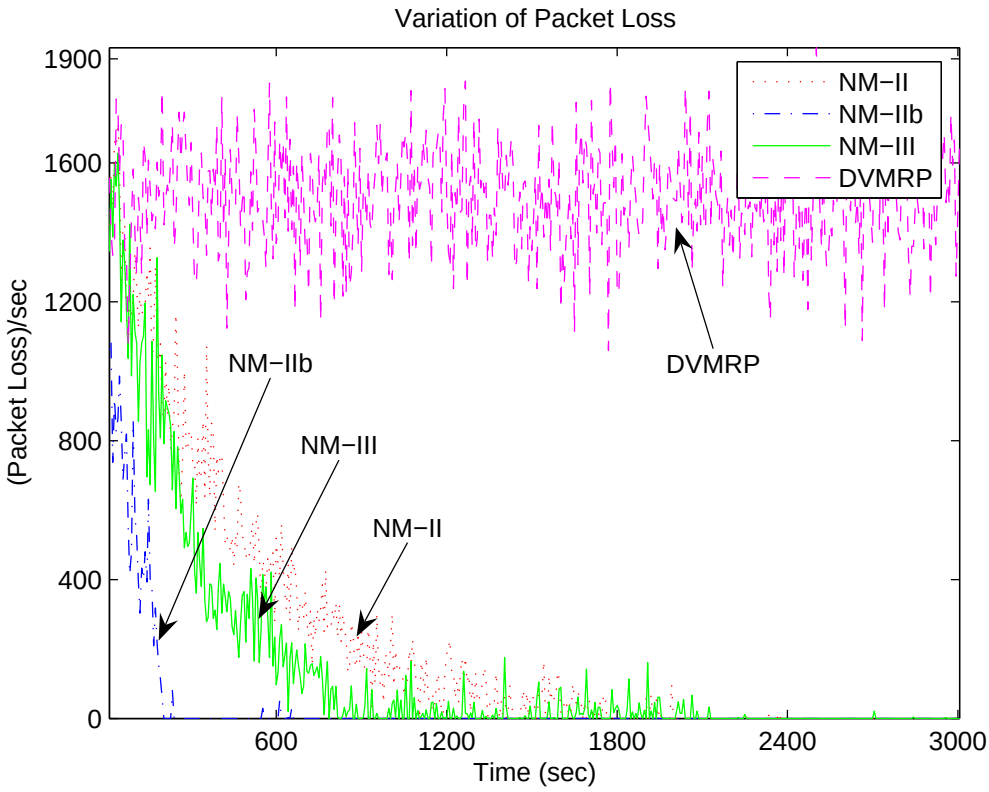
As shown in the figures, while our algorithm can reduce the network cost to a certain level under NM-I, it cannot completely eliminate packet losses and incurs a much higher overall cost compared to DVMRP. (The cost of NM-I model decreases to around 14.197 while it is around 7.45 for DVMRP.) The reason for this rather poor performance under NM-I is the lack of multicast functionality. Since we cannot create multicast trees, independent unicast sessions need to be set up to each destination from the sources and overlay nodes, ignoring the multicast nature of the traffic. This requires producing a separate copy of the same packet for each destination, even when these packets traverse many common links, and leads to severe link stress in the network.

It is clear from Figure 3.5 that our algorithm converges faster under NM-IIb than under any other model. This is due to the fact that we only need to optimize the overlay rates x_o^s instead of individual receiver rates $x_{o,d}^s$. Hence, the number of parameters that need to be updated in each iteration is much smaller than the other cases (6 versus 36).

In order to better understand the convergence rate of our algorithm under different network models we also simulate a scenario with 11 receivers for each source. The receiver sets are given by $D^1 = \{3, 4, 7, 8, 9, 10, 11, 12, 13, 16, 17\}$ and $D^5 = \{1, 2, 3, 4, 8, 9, 10, 12, 15, 16, 17\}$. The evolution of network cost and packet losses is plotted in Figure 3.6. We do not plot the performance of our algorithm un-



(a)



(b)

Figure 3.6: Network costs and packet losses with 11 receivers. (a) Network cost, (b) packet losses.

der NM-I for the ease of illustration (as it is not comparable to that under NM-II or NM-III). Clearly, as the number of receivers increases, the convergence of the algorithm under NM-II and NM-III becomes somewhat slower. Standard SPSA theory suggests that the convergence of an SPSA algorithm does not change significantly with the number of parameters. However, in our case sources perturb the system *independently* of others and they observe only *partial* network information as opposed to global information assumed in the standard SPSA model. We suspect that these are the main reasons for the slight performance degradation with increasing number of parameters in our case. Also, note that the algorithm does not suffer from slower convergence under NM-IIb with increasing number of receivers because the overlay node rates do not depend on the number of receivers. This points at another advantage of NM-IIb and suggests the robustness of its performance with respect to the number of receivers.

We note that in this scenario our algorithm effectively distributes the network load and eliminates the packet losses (except for under NM-I) with only three paths. We suspect that in many cases a small number of paths are sufficient to substantially improve the network performance, suggesting that only a limited number of core overlay nodes may be required in practice.

3.5.2 Second topology

Figure 3.7 shows the second topology we consider. It is a close approximation of Sprint backbone topology as reported in [44]. Compared to the first topology with

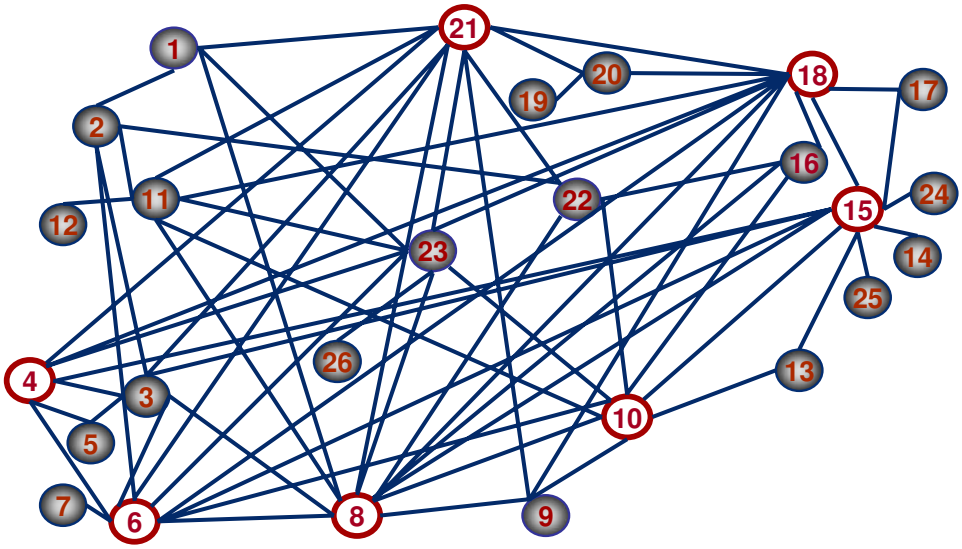


Figure 3.7: Network topology 2

an average node degree of 3.1667, the second topology is more densely connected and has an average node degree of 5.0769. As mentioned earlier in Chapter 1, recent findings suggest that many ISPs are in the process of increasing the node connectivity of their networks. In this subsection we are interested in understanding the effects of the density of the network on the relative performance of our algorithm.

The capacity of a link is 20 Mbps. There are three multicast sources and $\mathcal{S} = \{1, 9, 22\}$. Each source has 18 receivers. The receiver sets are given by $D^1 = \{2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 15, 16, 19, 20, 21, 22, 23, 25\}$, $D^9 = \{1, 2, 3, 4, 6, 7, 8, 10, 11, 13, 15, 16, 17, 18, 21, 22, 23, 24\}$ and $D^{22} = \{1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 15, 16, 17, 20, 21, 23, 26\}$. Nodes 10 and 23 are selected as additional core overlay nodes.[7] This gives $O^1 = \{1, 10, 23\}$, $O^9 = \{9, 10, 23\}$ and $O^{22} = \{22, 10, 23\}$, and each source-destination pair has three paths. Each source generates Poisson traffic

[7]Since both of the core nodes are also destination nodes for all three sources, under NM-I there are only two paths from the sources to these nodes.

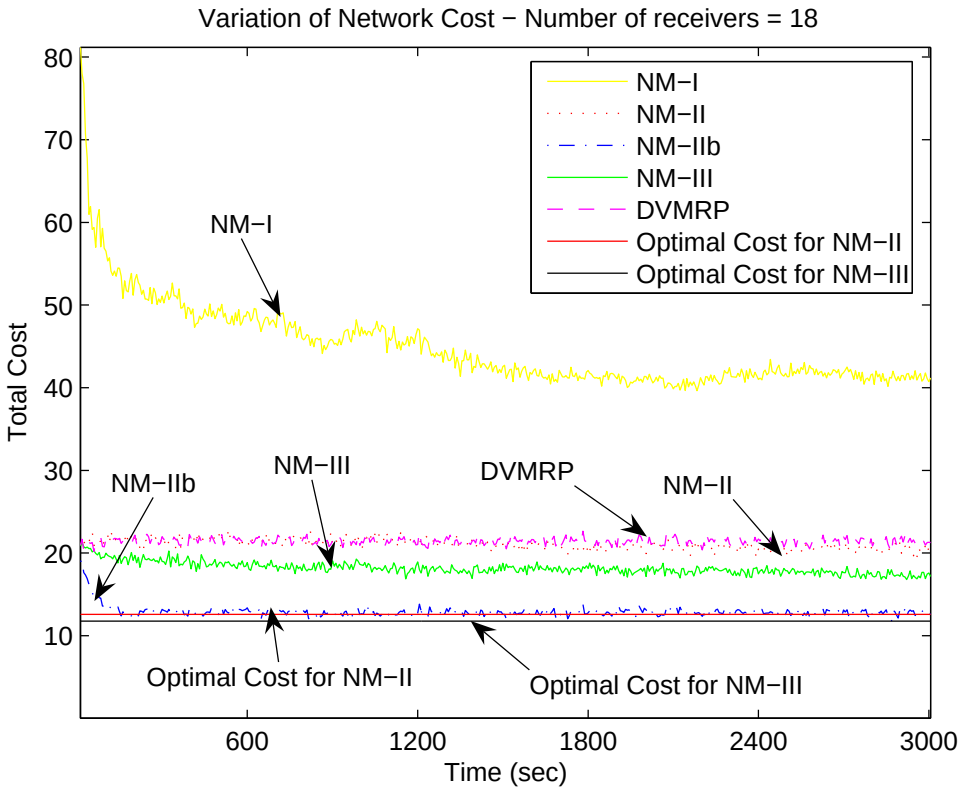
with an average rate of 10 Mbps.

Decreasing step sizes

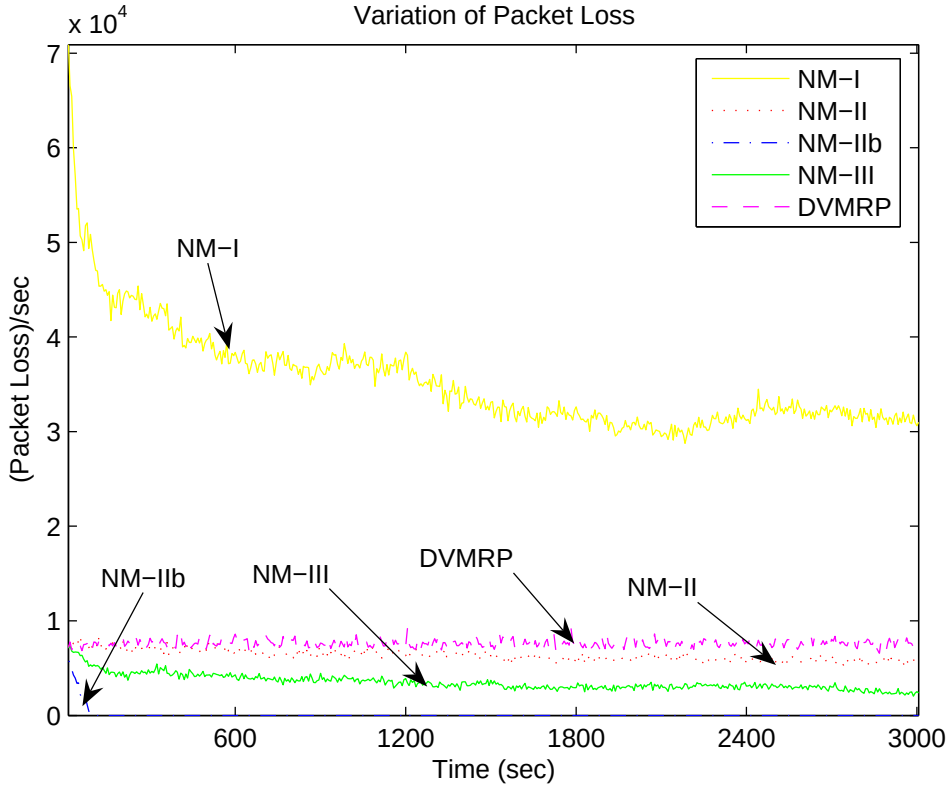
In this subsection we adopt the same decreasing step sizes used in the previous subsection and compare their relative performance. Figure 3.8 shows the evolution of network cost and packet loss rate. Similar to the previous case, our algorithm under NM-I experiences high link stress due to the lack of multicast capability and performs worse than DVMRP. Also, it is worth noting that the convergence of the algorithm under both NM-II and NM-III is considerably slower than under NM-IIb due to a relatively large number of receivers. The optimal cost values under NM-II and NM-III models are 12.5875 and 11.7612, respectively, and hence are close as in the previous case. This again suggests that much of the benefits can be obtained with the simpler NM-II (or NM-IIb) without the need to implement smart routers. Finally, we again see that three paths per receiver is sufficient to successfully distribute the traffic even when the DVMRP algorithm cannot eliminate the packet losses.

Constant step size

In this subsection we fix the step size $a_s(k) = a$ for all $s \in \mathcal{S}$ and $k = 0, 1, \ldots$, and compare the system performance for different values of a and against the decreasing step size case. Figure 3.9 plots the evolution of network cost and packet losses under NM-IIb for different values of step sizes. The values in parentheses



(a)



(b)

Figure 3.8: Network costs and packet losses - Poisson traffic source. (a) Network cost, (b) packet losses.

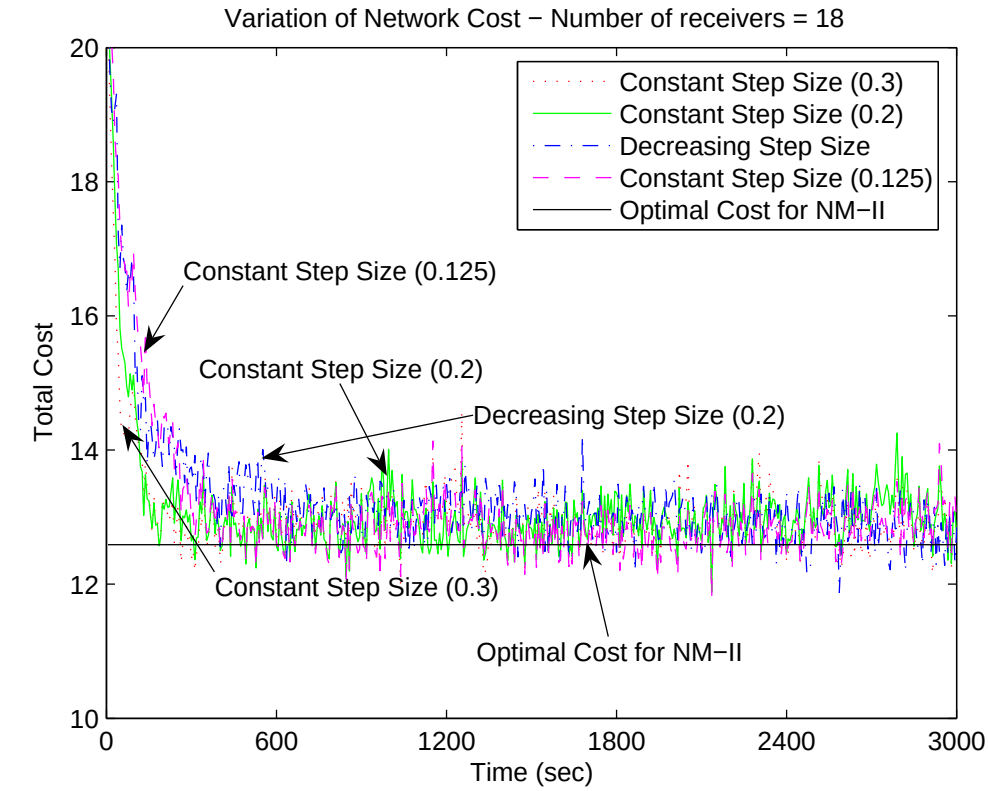
denote the step size. The figure clearly shows that when the step size is fixed the network cost oscillates slightly above the optimal value. Moreover, the initial convergence rate is better for several values of the fixed step size as the algorithm is able to reach a small neighborhood of the optimal operating points faster than with decreasing step sizes. Since this neighborhood is small and a constant step size policy is more robust in the presence of network fluctuations and can track dynamical changes in the network, this result suggests that the algorithm with a well chosen fixed step size may be preferred in practice.

MMPP sources

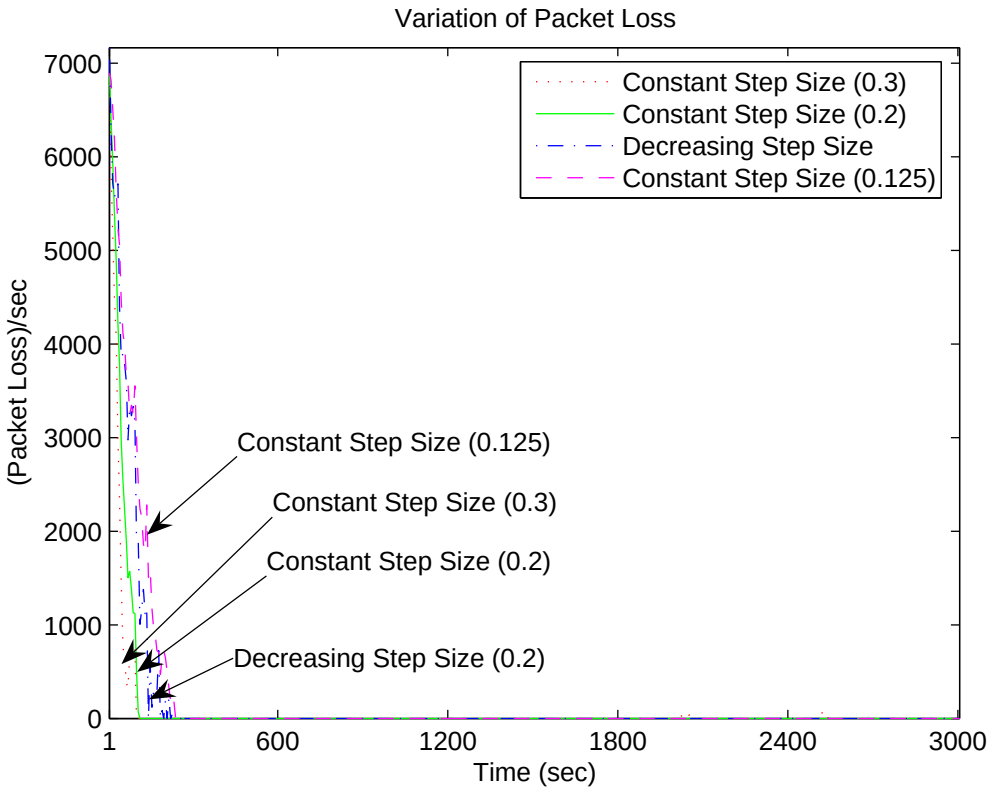
In the last set of experiments we adopt a different source model to simulate VBR video traffic.[8] It is shown in [21] that long-range dependence is not a crucial property in determining the buffer behavior of VBR video traffic and a Markov chain traffic model can be used to estimate the buffer occupancy well. Following this observation, we select a Markov Modulated Poisson Process (MMPP) as an alternative source model. We assume that each MMPP source with an average rate of 10 Mbps consists of 128 ON-OFF mini-sources with a mean ON period of 350 ms and a mean OFF period of 650 ms. We employ a decreasing step size policy for the simulation as in subsection 3.5.2.

From Figure 3.10 we see that the performance of the algorithm in each network model is similar to the Poisson traffic source case except for the higher variance

[8]Due to source encoding, it is possible to shape the incoming multicast traffic at the source nodes. However, this issue is ignored in the simulation studies.



(a)



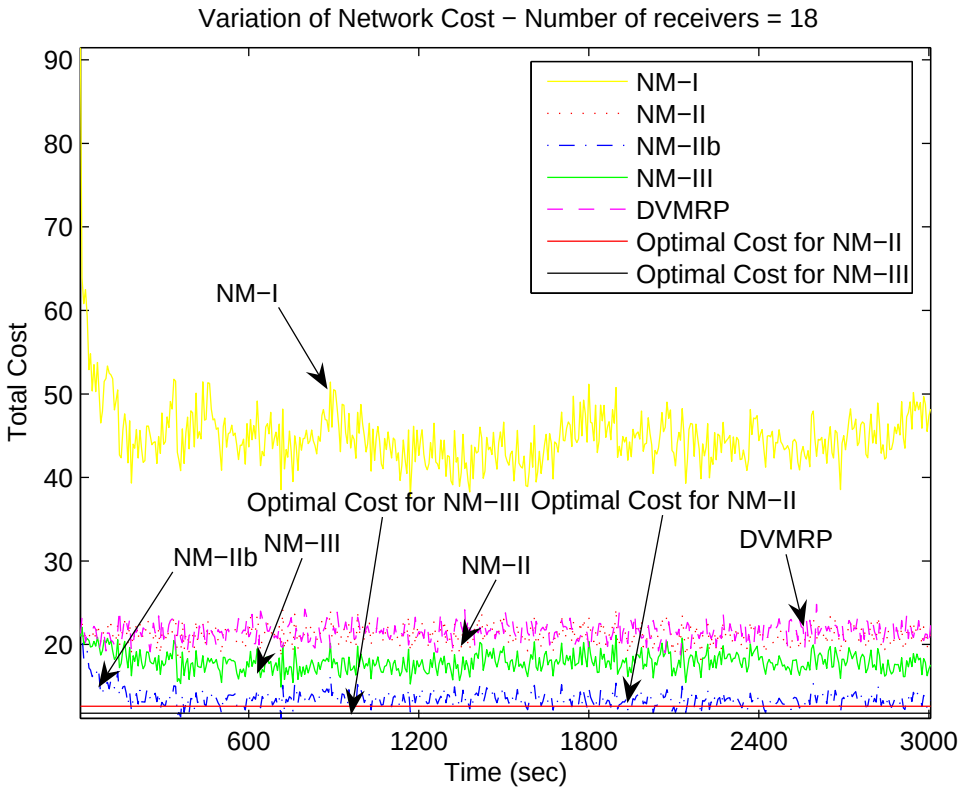
(b)

Figure 3.9: Network cost and packet drops with fixed step sizes and decreasing step sizes.

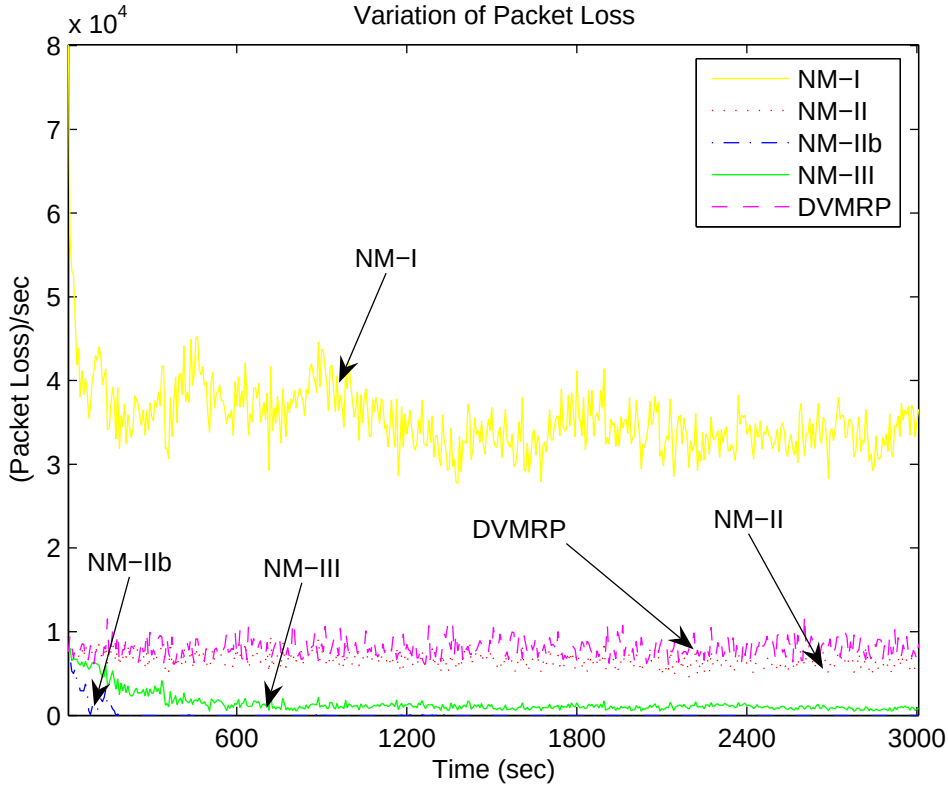
observed around the mean values. This is expected since the packet rates at the sources fluctuate around some mean value from the adopted traffic model.

3.6 Conclusion

We studied the issue of multipath routing of both unicast and multicast traffic where the gradient of network cost is not available and needs to be estimated using noisy measurements. We proposed an optimal routing algorithm based on the idea of simultaneous perturbation stochastic approximation with provable convergence properties. Three different network models (NM-I, II, and III) were considered to evaluate its performance and to quantify the benefits of additional functionality/intelligence in the underlying IP network. In NM-III we established a routing framework that generalizes the multiple distribution trees to a more general multiple path scenario where each destination can receive packets at a different rate from a multicast tree. The need for complicated bookkeeping at the sources and intermediate IP routers is handled by employing Digital Fountain coding. We proved the convergence properties of the proposed algorithm both with decreasing step sizes and with a fixed step size. Further, our simulation studies show that while basic IP multicast functionality in NM-II is crucial for better performance, additional functionalities introduced in NM-III provide only marginal benefits in relation to required complexity.



(a)



(b)

Figure 3.10: Network cost and packet losses - MMPP source. (a) Network cost, (b) packet losses.

3.7 Proof of Theorem 3.4.2

The basic approach we follow in the proof is similar to that in [27, pp. 127-131] and [19]. A main difference between the proof in [19] and the current proof stems from the fact that the cost function is no longer continuously differentiable with respect to the rate assignment although the convexity is preserved. This is because of the max operator in the link cost functions in (3.1) - (3.3). Here we will show that the same convergence holds even when the cost function is not differentiable, borrowing tools from the convex analysis and the concept of subgradient.

Collecting the terms of (3.7) for all sources, we have the following update rule for the system:

$$\boldsymbol{x}(k+1) = \Pi_{\Theta}[\boldsymbol{x}(k) - \boldsymbol{A}(k)\hat{\boldsymbol{g}}(k)] \ , \tag{3.11}$$

where $\hat{\boldsymbol{g}}(k) := (\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k), s \in \mathcal{S})$, $\boldsymbol{A}(k)$ is an $N \times N$ diagonal matrix with the diagonal entries $\boldsymbol{A}_{ii}(k)$ given by the step sizes $a_s(k)$ of corresponding sources. This is different from the standard stochastic approximation algorithms that have a scalar step size.

In the rest of the proof we closely follow the same outline in [27] with the gradient $\boldsymbol{\nabla} C(\boldsymbol{x})$ replaced by a subgradient $\boldsymbol{sg}(\boldsymbol{x})$. Unless stated otherwise $\boldsymbol{E}_k(\cdot)$ represents $\mathbf{E}[\cdot|\mathcal{F}_k]$. Rewrite (3.11) in the following form

$$\begin{aligned}
\boldsymbol{x}(k+1) &= \Pi_{\Theta}[\boldsymbol{x}(k) - \boldsymbol{A}(k)\hat{\boldsymbol{g}}(k)] \\
&= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k), \\
&= \boldsymbol{v}(k) + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)
\end{aligned} \tag{3.12}$$

where

$$\begin{aligned}
\boldsymbol{v}(k) &= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)], \\
\boldsymbol{\varrho}(k) &= \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)) - \hat{\boldsymbol{g}}(k),\ \boldsymbol{b}(k) = \boldsymbol{sg}(\boldsymbol{x}(k)) - \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)), \\
\boldsymbol{\tau}(k) &= \Pi_\Theta[\boldsymbol{v}^\gamma(k)] - \boldsymbol{v}^\gamma(k), \\
\boldsymbol{v}^\gamma(k) &= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)]I(k), \\
\boldsymbol{\phi}(k) &= \boldsymbol{v}^\gamma(k) - \boldsymbol{v}(k) + \big(\Pi_\Theta[\boldsymbol{v}(k)] - \boldsymbol{x}(k)\big)\big(1 - I(k)\big),
\end{aligned}$$

$I(k)$ is an indicator function of the event $\{||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \leq \gamma(k)/2\}$, and $\{\gamma(k), k = 0, 1, \ldots\}$ is a sequence of positive real numbers such that (i) $\gamma(k) \to 0$ and (ii) $||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \leq \gamma(k)/2$ for all but a finite number of k w. p. 1. The following lemma is used to show the existence of such a sequence.

Lemma 3.7.1. *Under Assumptions **A1** - **A5**, the bias and error terms given by $\boldsymbol{b}(k)$ and $\boldsymbol{\varrho}(k)$, respectively, satisfy the following:*

(a) $\boldsymbol{b}(k) \to 0$ w. p. 1 , and

(b) $\sum_{k=0}^{\infty} \boldsymbol{E}_k(||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)||^2) < \infty$ w. p. 1.

Proof: Please see Section 3.9 for a proof. □

Now we prove the existence of the sequence $\{\gamma(k), k = 0, 1, \ldots\}$ that satisfies the given conditions.

Since $\sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i)$, $j \geq k$, is a martingale and $\sum_{i=k}^{\infty} \boldsymbol{E}_i(||\boldsymbol{A}(i)\boldsymbol{\varrho}(i)||^2) < \infty$ from Lemma 3.7.1, by the Doob-Kolmogorov inequality ([17, p. 309]), for every $\epsilon > 0$,

we have

$$P\Big(\sup_{j\geq k}\Big|\Big|\sum_{i=k}^{j}\boldsymbol{A}(i)\boldsymbol{\varrho}(i)\Big|\Big|\geq\epsilon\Big)\leq\frac{1}{\epsilon^2}\sum_{i=k}^{\infty}\boldsymbol{E}_i(||\boldsymbol{A}(i)\boldsymbol{\varrho}(i)||^2)\ .$$

Since the right-hand side of the equation goes to zero as $k\to\infty$, it follows that

$$\lim_{k\to\infty}P\Big(\sup_{j\geq k}\Big|\Big|\sum_{i=k}^{j}\boldsymbol{A}(i)\boldsymbol{\varrho}(i)\Big|\Big|\geq\epsilon\Big)=0\ ,$$

and the existence of the sequence $\{\gamma(k),k=0,1,\ldots\}$ is guaranteed.

Let $\boldsymbol{X^0}(t)$ be the following continuous time interpolation of $\boldsymbol{x}(k)$.

$$\boldsymbol{X^0}(t):=\frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{x}(k)+\frac{t-t_k}{\hat{a}(k)}\boldsymbol{x}(k+1)\ ,\ t\in[t_k,t_{k+1}) \tag{3.13}$$

where $t_k=\sum_{i=0}^{k-1}\hat{a}(i)$.[9] We will show that the trajectory of this continuous time process follows that of the differential inclusion to be defined shortly, and the path of this differential inclusion will be shown to converge to a point in the set of solutions of the optimization problem we consider.

Let

$$\boldsymbol{B^0}(t_k):=\sum_{i=0}^{k-1}\boldsymbol{A}(i)\boldsymbol{b}(i),\quad \boldsymbol{M^0}(t_k):=\sum_{i=0}^{k-1}\boldsymbol{A}(i)\boldsymbol{\varrho}(i),$$
$$\mathbf{T^0}(t_k):=\sum_{i=0}^{k-1}\boldsymbol{\tau}(i),\qquad \boldsymbol{\Phi^0}(t_k):=\sum_{i=0}^{k-1}\boldsymbol{\phi}(i)$$

and the corresponding piecewise linear interpolations for $t\in(t_k,t_{k+1})$ are defined similarly as in (3.13).

[9]Recall that $\hat{a}(k)=\max_{s\in\mathcal{S}}a_s(k)$.

We rewrite $\boldsymbol{X^0}(t)$ using the relation (3.12) as follows.

$$
\begin{aligned}
&\boldsymbol{X^0}(t) \\
&= \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\Big(\boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) \\
&\qquad\qquad +\boldsymbol{\varrho}(k) + \boldsymbol{b}(k)] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\Big) \\
&= \boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\Big(\boldsymbol{A}(k)\big[- \boldsymbol{sg}(\boldsymbol{x}(k)) \\
&\qquad\qquad +\boldsymbol{\varrho}(k) + \boldsymbol{b}(k)\big] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\Big) \\
&= \boldsymbol{x}(0) + \sum_{i=0}^{k-1}\Big(\boldsymbol{A}(i)[-\boldsymbol{sg}(\boldsymbol{x}(i)) + \boldsymbol{b}(i) + \boldsymbol{\varrho}(i)] \\
&\quad +\boldsymbol{\tau}(i) + \boldsymbol{\phi}(i)\Big) + \frac{t-t_k}{\hat{a}(k)}\Big(\boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) \\
&\quad +\boldsymbol{b}(k)] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\Big) \\
&= \boldsymbol{x}(0) + \boldsymbol{B^0}(t) + \boldsymbol{M^0}(t) - \sum_{i=0}^{k-1}\hat{a}(i)\boldsymbol{sg}(\boldsymbol{x}(i)) \\
&\quad -(t-t_k)\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{Z^0}(t) + \mathbf{T^0}(t) + \boldsymbol{\Phi^0}(t) \\
&= \boldsymbol{x}(0) + \boldsymbol{B^0}(t) + \boldsymbol{M^0}(t) + \boldsymbol{H^0}(t) + \boldsymbol{Z^0}(t) \\
&\quad +\mathbf{T^0}(t) + \boldsymbol{\Phi^0}(t)\ ,
\end{aligned}
$$

where

$$
\begin{aligned}
\boldsymbol{Z^0}(t) &= \sum_{i=0}^{k-1}\big(\hat{a}(i) - \boldsymbol{A}(i)\big)\boldsymbol{sg}(\boldsymbol{x}(i)) \\
&\qquad +\frac{t-t_k}{\hat{a}(k)}\big(\hat{a}(i) - \boldsymbol{A}(i)\big)\boldsymbol{sg}(\boldsymbol{x}(k))\ , \\
\boldsymbol{H^0}(t) &= -\int_0^t \boldsymbol{sg}(\bar{\boldsymbol{x}}(s))ds\ ,\ \text{and} \\
\bar{\boldsymbol{x}}(s) &= \boldsymbol{x}(k)\ ,\ s \in [t_k, t_{k+1})\ .
\end{aligned}
$$

We are interested in the tail properties of the interpolated processes. To this

end we introduce the following time shifted and centered processes.

$$\boldsymbol{X^k}(t):=\boldsymbol{x}(k) + \boldsymbol{B^k}(t) + \boldsymbol{M^k}(t) + \boldsymbol{H^k}(t) + \boldsymbol{Z^k}(t)$$
$$+\mathbf{T^k}(t) + \boldsymbol{\Phi^k}(t) \ , \tag{3.14}$$

where $\boldsymbol{B^k}(t) = \boldsymbol{B^0}(t_k + t) - \boldsymbol{B^0}(t_k)$, $\boldsymbol{M^k}(t)$, $\mathbf{T^k}(t)$, $\boldsymbol{\Phi^k}(t)$ and $\boldsymbol{Z^k}(t)$ are defined similarly, and $\boldsymbol{H^k}(t) = -\int_0^t \boldsymbol{sg}(\bar{\boldsymbol{x}}(t_k + s))ds$.

We now show that every process on the right hand of (3.14) is equicontinuous and hence $\boldsymbol{X^k}(t)$ is also equicontinuous. First, from Lemma 3.7.1 and the definition of $\{I(k), k = 0, 1, \ldots\}$ one can show that there is a null set Ω_0 such that for each outcome $\omega \notin \Omega_0$, the processes $\boldsymbol{B^k}(t)$, $\boldsymbol{M^k}(t)$ and $\boldsymbol{\Phi^k}(t)$ converge to zero uniformly on finite intervals as $k \to \infty$. Second, under Assumptions **A5** and **A1** (in particular, $\partial C(\boldsymbol{x})$ is bounded for all $\boldsymbol{x} \in \Theta$, which implies that there exists a finite bound B on $\boldsymbol{sg}(\boldsymbol{x}(k))$) it is easy to verify that $\boldsymbol{Z^k}(t) \to 0$ as $k \to \infty$. Hence, $\boldsymbol{B^k}(t)$, $\boldsymbol{M^k}(t)$, $\boldsymbol{\Phi^k}(t)$, and $\boldsymbol{Z^k}(t)$ converge to zero uniformly on each bounded interval in $(-\infty, \infty)$ as $k \to \infty$, and thus are equicontinuous [27, p. 101]. The equicontinuity of $\boldsymbol{H^k}(t)$ follows from the same argument in the proof of Proposition 1 in [20] because of the existence of a finite bound B on $\boldsymbol{sg}(\boldsymbol{x})$ mentioned before. The equicontinuity of $\mathbf{T^k}(t)$ can shown by the same argument in the proof of Theorem 5.2.1 in [27, p. 128].

Since the processes on the right-hand side of (3.14) are equicontinuous, so is $\boldsymbol{X^k}(t)$. Therefore, Arzela-Ascoli Theorem [27, p. 100] tells us that, for every $\omega \notin \Omega_0$, there exists a subsequence $\{k_j, j = 1, 2, \ldots\}$ such that $\{\boldsymbol{X}^{k_j}(\omega, .), \boldsymbol{H}^{k_j}(\omega, .), \mathbf{T}^{k_j}(\omega, .)\}$ converges to some $\{\boldsymbol{X}(\omega, .), \boldsymbol{H}(\omega, .), \mathbf{T}(\omega, .)\}$ uniformly and, following the

same argument in the proof of Proposition 1 in [20], we can write

$$\boldsymbol{X}(\omega,t) \;=\; \boldsymbol{X}(\omega,0)+\boldsymbol{H}(\omega,t)+\mathbf{T}(\omega,t)\;. \tag{3.15}$$

The second term on the right-hand side $\boldsymbol{H}(\omega,t) = \int_0^t \tilde{\boldsymbol{h}}(\omega,s)ds$, where $\tilde{\boldsymbol{h}}(\omega,s) \in -\partial C(\boldsymbol{X}(\omega,s))$ [20]. The third term $\boldsymbol{T}(\omega,t) = \int_0^t \tilde{\boldsymbol{\tau}}(\omega,s)ds$ with $\tilde{\boldsymbol{\tau}}(\omega,s) \in -V(\boldsymbol{X}(\omega,s))$ for almost all s [27, pp. 128-129], where $V(\boldsymbol{x})$ is the convex cone generated by the set of outward normals $\{\boldsymbol{y} : \boldsymbol{y} = \boldsymbol{\nabla q_i}(\boldsymbol{x}),\ \text{s.t.}\ \boldsymbol{q_i}(\boldsymbol{x}) = 0\}$ and $\boldsymbol{q_i}(\cdot)$ are the constraints of the optimization problem.

From (3.15) it is plain that the limit $\boldsymbol{X}(\omega,.)$ of any convergent subsequence satisfies the differential inclusion:

$$\dot{\boldsymbol{X}} \in -\partial C(\boldsymbol{X}) + \mathbf{T}\;,\;\; \mathbf{T}(t) \in -V(\boldsymbol{X}(t)). \tag{3.16}$$

This tells us [20] that the limit $\boldsymbol{X^k}(\omega,\cdot)$ converges to the stationary set of points of (3.16) in Θ w. p. 1. We denote this set of stationary points where $0 \in -\partial C(\boldsymbol{x}) + \boldsymbol{\tau}$ by S_Θ. Since $C(\cdot)$ is a convex function and Θ is a nonempty convex set, any point in S_Θ attains the minimum of $C(\cdot)$, and this completes the proof of the theorem.

3.8 Proof of Theorem 3.4.4

We use similar arguments used in the proof of Theorems 8.2.1 and 8.2.5 of [27, pp. 251-254 and 261-262] in our proof. First, the existence of a convergent subsequence follows from the fact that the uniform integrability of $\{\hat{\boldsymbol{g}}^n(k); n, k\}$ in Assumption **A8** implies tightness of $\{\boldsymbol{X}^n(a_nq_n+\cdot), \boldsymbol{\Upsilon}^n(a_nq_n+\cdot)\}$ [27, p. 253], which is a sufficient condition for the existence of a convergent subsequence.

Following the argument in the proof of Lemma 3.7.1, for sufficiently small step size a_n we can replace the projection of $\boldsymbol{x}^n(k)+\xi^n\boldsymbol{\Delta}(k)$ with the linear approximation in (3.17). Thus, from (3.21) we have

$$\boldsymbol{E}_k(\hat{\boldsymbol{g}}^n(k)) \;=\; \overline{\boldsymbol{sg}}(\boldsymbol{x}^n(k)) + O(a_n) + \delta^n(k)$$

First, from the argument in the proof of Lemma 3.7.1 (in particular, eq. (3.20)) one can show that $\delta^n(k) \to 0$ as $n \to 0$ (hence $a_n \to 0$). Second,

$$\lim_{m,\overline{m},a} \frac{1}{m} \sum_{i=\overline{m}}^{\overline{m}+m-1} O(a) = 0$$

if we take $\overline{m} = m$ and $a = m^{-2}$ with $m \to \infty$.

Now since the mapping from $\boldsymbol{x} \in \Theta$ to the subdifferential $\partial C(\boldsymbol{x})$ is upper semicontinuous, from [27, pp. 261-262], the limit process $\{\boldsymbol{X}(\cdot), \boldsymbol{\Upsilon}(\cdot)\}$ satisfies

$$\dot{\boldsymbol{X}} \in \partial C(\boldsymbol{X}) + \boldsymbol{\Upsilon},\; \boldsymbol{\Upsilon}(t) \in -V(\boldsymbol{X}(t))$$

for almost all ω and t .

Similarly as in the proof of Theorem 3.4.2, this implies that the limit $\boldsymbol{X^k}(\cdot)$ converges to the stationary set of points of (3.16) in Θ. This completes the proof.

3.9 Proof of Lemma 3.7.1

Let $\bar{\boldsymbol{\Delta}}_s(k)$ be an $N \times 1$ vector, where values of entries corresponding to those of source s are $\Delta_{s,i}(k)$ and zero otherwise. Hence, $\boldsymbol{\Delta}(k) = \sum_{s\in S} \bar{\boldsymbol{\Delta}}_s(k) = (\Delta_{s,i}, s \in S, i \in 1, 2, \cdots, N_s \cdot |D^s|)$. Similarly, $\boldsymbol{u_s}$ is an $N \times 1$ vector, where the values of entries corresponding to those of source s are one and zero otherwise.

First, we show that there exists a finite K_1 such that for all $k \geq K_1$ we can rewrite $C_s^{\pm}(k)$ as

$$\begin{aligned} C_s^-(k) &= \Lambda_s(\boldsymbol{x}(k)) \text{ and} \\ C_s^+(k) &= \Lambda_s\left(\boldsymbol{x}(k) + \sum_{s' \in S} \xi_{s'}(k)\tilde{\boldsymbol{\Delta}}_{s'}(k)\right) , \end{aligned}$$

where

$$\tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) = \boldsymbol{\Delta}_{\boldsymbol{s'}}(k) - \frac{\sum_{j=1}^{N_{s'}} \Delta_{s',j}(k)}{N_{s'}} \boldsymbol{u}_{\boldsymbol{s'}} .$$

Given any $N_s \times 1$ vector $\boldsymbol{\vartheta}$, the solution of the minimization problem

$$\begin{aligned} &\min_{\boldsymbol{\eta}} \ ||\boldsymbol{\vartheta} - \boldsymbol{\eta}||^2 \\ &\text{s. t. } \boldsymbol{\eta}^T \boldsymbol{u} = r_s \end{aligned}$$

is given by

$$\eta_i = \vartheta_i + \frac{r_s - \sum_{j=1}^{N_s} \vartheta_j}{N_s} , \tag{3.17}$$

where $\boldsymbol{u} = [1, 1, \cdots, 1]^T$. Obviously, if $\eta_i \geq 0$ for all i, this solution is equivalent to the L_2 projection. Here for the purpose of temporary perturbation we replace (3.6) with a non-negativity constraint. Thus, the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ can be calculated using (3.17) if

$$x_{s,i}(k) + \xi_s(k)\left(\Delta_{s,i}(k) - \frac{\sum_{j=1}^{N_s}(\Delta_{s,j}(k))}{N_s}\right) \geq 0 . \tag{3.18}$$

Recall that $\Delta_{s,i}(k)$ is bounded by α from Assumption **A2**. Hence, (3.18) holds if $\xi_s(k) \leq \frac{\min_j\{x_{s,j}(k)\}}{2\alpha}$. From (3.6) we know $\frac{\min_j\{x_{s,j}(k)\}}{2\alpha} \geq \frac{\epsilon}{2\alpha}$. Since $\xi_s(k) \to 0$, there exists finite K_1 such that $\xi_s(k) \leq \frac{\epsilon}{2\alpha}$ for all $k \geq K_1$. Therefore, (3.17) can be used to compute the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ for sufficiently large $k \geq K_1$.

Define for $k \geq K_1$

$$\begin{aligned} G_k^s &:= \frac{C_s^+(k) - C_s^-(k)}{\xi_s(k)} \\ &= \frac{\Lambda_s\left(\boldsymbol{x}(k) + \sum_{s' \in S} \xi_{s'}(k)\tilde{\boldsymbol{\Delta}}_{s'}(k)\right) - \Lambda_s(\boldsymbol{x}(k))}{\xi_s(k)} \\ &= \frac{\Lambda_s\left(\boldsymbol{x}(k) + \xi_s(k)\hat{\boldsymbol{\Delta}}(k)\right) - \Lambda_s(\boldsymbol{x}(k))}{\xi_s(k)}, \end{aligned}$$

where

$$\hat{\boldsymbol{\Delta}}(k) = \sum_{s' \in S} \frac{\xi_{s'}(k)}{\xi_s(k)} \tilde{\boldsymbol{\Delta}}_{s'}(k)$$

Definition 3.9.1 ([20]). *Suppose that $h : \mathbb{R}^r \to \mathbb{R}$ is a real-valued convex function on $\mathbb{R}^r$. The one-sided directional derivative of h at $\boldsymbol{x}$ with respect to a vector $\boldsymbol{y}$ is defined to be*

$$h'(\boldsymbol{x}; \boldsymbol{y}) \;=\; \lim_{\lambda \downarrow 0} \frac{h(\boldsymbol{x} + \lambda \boldsymbol{y}) - h(\boldsymbol{x})}{\lambda} \;.$$

For each vector $\boldsymbol{y}$ the directional derivative satisfies

$$h'(\boldsymbol{x}; \boldsymbol{y}) \;=\; \max_{sg(\boldsymbol{x}) \in \partial h(\boldsymbol{x})} sg(\boldsymbol{x})^T \boldsymbol{y} \;.$$

In other words, the directional derivative is the maximum of the inner products $< sg(\boldsymbol{x}), \boldsymbol{y} >$ over $\partial h(\boldsymbol{x})$. We denote the set of subgradients that achieve the maximal inner product by $\partial h_{\boldsymbol{y}}(\boldsymbol{x}) \subseteq \partial h(\boldsymbol{x})$, i.e., $\partial h_{\boldsymbol{y}}(\boldsymbol{x}) = \arg\max_{sg(\boldsymbol{x}) \in \partial h(\boldsymbol{x})} sg(\boldsymbol{x})^T \boldsymbol{y}$.

Let $\Lambda_s'(\boldsymbol{x}(k)\,; \hat{\boldsymbol{\Delta}}(k))$ (resp. $C'(\boldsymbol{x}(k)\,; \hat{\boldsymbol{\Delta}}(k))$) be the one-sided directional derivative of Λ_s (resp. C) at $\boldsymbol{x}(k)$ with respect to vector $\hat{\boldsymbol{\Delta}}(k)$. Since $\Lambda_s(\cdot)$ is convex and continuous, we can show using [20, Lemma 1] that, for any $\epsilon > 0$, there exists finite

K_2 such that, for all $k \geq K_2$,

$$|\Lambda'_s(\boldsymbol{x}(k); \hat{\boldsymbol{\Delta}}(k)) - G^s_k| < \frac{\epsilon}{2} \qquad \text{w. p. 1.}$$

From above, for every $\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k)) \in \partial\Lambda_{s,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$, we have

$$\begin{aligned}\Lambda'_s(\boldsymbol{x}(k); \hat{\boldsymbol{\Delta}}(k)) &= \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \hat{\boldsymbol{\Delta}}(k) \\ &= \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s' \in S} \frac{\xi_{s'}(k)}{\xi_s(k)} \tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) \ . \end{aligned} \tag{3.19}$$

Since $\frac{\xi_{s'}(k)}{\xi_s(k)} \to 1$, there exists finite $K_3 \geq \max(K_1, K_2)$ such that, for all $\epsilon > 0$ and $k \geq K_3$,

$$\left| \frac{C^+_s(k) - C^-_s(k)}{\xi_s(k)} - \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s' \in S} \tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) \right| < \epsilon \qquad \text{w. p. 1.} \tag{3.20}$$

Consequentially, for all $k \geq K_3$,

$$\begin{aligned}&\boldsymbol{E}_k(\hat{g}_{s,m}(k)) \\ &= \frac{N_s}{N_s - 1} \boldsymbol{E}_k \left(\frac{C^+_s(k) - C^-_s(k) + \mu^+_s(k) - \mu^-_s(k)}{\xi_s(k) \Delta_{s,m}(k)} \right) \\ &= \frac{N_s}{N_s - 1} \boldsymbol{E}_k \left(\frac{\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s' \in S} \tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) + \delta_1(k)}{\Delta_{s,m}(k)} \right) \\ &= \frac{N_s}{N_s - 1} \left(\frac{N_s - 1}{N_s} \overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k)) + O(\xi_s(k)) \right) + \bar{\delta}_1(k) \\ &= \overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k)) + O(\xi_s(k)) + \bar{\delta}_1(k) \end{aligned} \tag{3.21}$$

where $\overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k))$ is the entry of $\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))$ corresponding to $\boldsymbol{x}_{s,m}(k)$,

$$\delta_1(k) = \frac{C^+_s(k) - C^-_s(k)}{\xi_s(k)} - \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s' \in S} \tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) \ ,$$

and

$$\bar{\delta}_1(k) = \frac{N_s}{N_s - 1} \boldsymbol{E}_k \left(\frac{\delta_1(k)}{\Delta_{s,m}(k)} \right) \ .$$

It is now plain to see that $\lim_{k\to\infty} \bar{\delta}_1(k) \to 0$ w. p. 1 from (3.20) and hence

$$\boldsymbol{E}_k(\hat{g}_{s,m}(k)) - \overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k)) \to 0 \qquad \text{w. p. 1.} \tag{3.22}$$

According to [6, Proposition 4.2.4, p. 232], if a real-valued function h is a sum of a set of convex functions $\{h_1, \ldots, h_n\}$, i.e., $h(\boldsymbol{x}) = \sum_{i=1}^{n} h_i(\boldsymbol{x})$, the subdifferential of h at $\boldsymbol{x}$ is given by

$$\begin{aligned}\partial h(\boldsymbol{x}) &= \sum_{i=1}^{n} \partial h_i(\boldsymbol{x}) \\ &:= \left\{ \sum_{i=1}^{n} sg^i(\boldsymbol{x}) : \ sg^i(\boldsymbol{x}) \in \partial h_i(\boldsymbol{x}) \text{ for all } i \in \{1, \ldots, n\} \right\}.\end{aligned}$$

This tells us that, for each $s \in \mathcal{S}$, the subgradient $\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))$ in (3.19) can be written as

$$\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k)) \ = \ \sum_{l \in L^s} \overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k)) \ ,$$

where $\overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k)) \in \partial C_{l,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$. In addition, this implies that any summation $\sum_{l\in\mathcal{L}} \overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k))$ with $\overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k)) \in \partial C_{l,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$ for all $l \in \mathcal{L}$ is a subgradient in $\partial C_{\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$.

For each $l \in \mathcal{L}$, choose a subgradient $\boldsymbol{sg}^{l\star}(\boldsymbol{x}(k)) \in \partial C_{l,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$ and let $\overline{\boldsymbol{sg}}^{\star}_s(\boldsymbol{x}(k)) = \sum_{l\in L^s} \boldsymbol{sg}^{l\star}(\boldsymbol{x}(k))$, $s \in \mathcal{S}$, and $\boldsymbol{sg}^{\star}(\boldsymbol{x}(k)) = \sum_{l\in\mathcal{L}} \boldsymbol{sg}^{l\star}(\boldsymbol{x}(k))$. Then, from the above argument, it is clear that $\overline{\boldsymbol{sg}}^{\star}_s(\boldsymbol{x}(k))$ satisfies (3.19) for all $s \in \mathcal{S}$. Moreover, if we denote by $s(m)$ the source corresponding to the m-th entry of $\boldsymbol{x}$, then the m-th entry of $\boldsymbol{sg}^{\star}(\boldsymbol{x}(k))$ equals the m-th entry of $\overline{\boldsymbol{sg}}^{\star}_{s(m)}(\boldsymbol{x}(k))$. Therefore, from (3.22) this proves claim (i) of the lemma that there exists a sequence of subgradients $\{\boldsymbol{sg}(\boldsymbol{x}(k)), k = 0, 1, \ldots\}$ such that $\boldsymbol{b}(\boldsymbol{x}(k)) := \boldsymbol{sg}(\boldsymbol{x}(k)) - \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k))$ goes to 0 w. p. 1.

From the assumption that $\mathbf{E}\left[\mu_s^+(k) - \mu_s^-(k)|\mathcal{F}_k\right] = 0$ and using the independence of $\mu_s^{\pm}(k)$ and $\boldsymbol{\Delta_s}(k)$, we can bound the second moment of $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ as follows:

$$\begin{aligned}
&\boldsymbol{E}_k((\hat{g}_{s,i}(k))^2) \\
&= \boldsymbol{E}_k\left(\left(\frac{C^+(k) - C^-(k) + \mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2\right) \\
&= \boldsymbol{E}_k\left(\left(\frac{C^+(k) - C^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2 + \left(\frac{\mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2\right)
\end{aligned} \tag{3.23}$$

After some algebra using the bounds on $\mathbf{E}\left[(\boldsymbol{\Delta_s}(k))^2\right]$, $\mathbf{E}\left[(\boldsymbol{\Delta_s}(k))^{-2}\right]$, and $\mathbf{E}\left[(\mu_s^{\pm}(k))^2\right]$, one can show that the first term in (3.23) is $O(1)$ and the second term is $O(\xi_s(k)^{-2})$. Note that $\boldsymbol{E}_k(\boldsymbol{\varrho}^2(k)) = \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)^2) - (\boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)))^2$. Hence, since $\sum_{k=0}^{\infty}\left(\frac{a_s(k)}{\xi_s(k)}\right)^2 < \infty$ from Assumption **A4**, it is easy to see that $\sum_{i=k}^{\infty}\boldsymbol{E}_k(||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)||^2) < \infty$ w. p. 1.

Chapter 4

Obtaining Better Paths through Overlay Node Selection

4.1 Introduction

We consider a network, where every node potentially has overlay capabilities. This allows us the possibility that every node can be used as an overlay node.[1]. However, in practice it is beneficial to activate the overlay capability only on a limited number of nodes due to the following reasons. First, much of the benefit from multipath routing can be obtained using only a subset of nodes. In fact, activating all overlay nodes may only create redundant alternate paths without providing any additional benefit. Moreover, the alternative paths established through overlay nodes tend to have higher delay compared to the default paths given by the underlying routing protocol due to generally larger hop counts as well as due to the processing time required to redirect the packets to the final destinations by the overlay application. Finally, the convergence rates of the proposed routing algorithms slow down as the number of paths per SD pair increases. This is actually in contradiction with the basic SPSA convergence results which, as discussed in Chapter 2, claim that the convergence of the SPSA algorithm does not depend on the size of the vector

[1]In this work we only consider single level overlays, *i.e.,* using at most one overlay node along each path established. However, it is easy to generalize the results to the general setting where one can establish paths with more than one overlay node along the path.

of input parameters. However, this result is not necessarily valid for the routing algorithms proposed in the previous chapters. This is because unlike the global cost information used in regular SPSA, each SD pair makes use of only the local network state information. As each SD pair independently perturbs their paths to converge under the SPSA model, they create additional noise to each other in measuring their local network state. In fact, simulation results presented in the previous chapters support this argument although the corresponding analytical result has not been established.

In the light of all the above, in this chapter we consider activating the overlay capability on a limited number of network nodes that will allow us to effectively load balance the network traffic.

4.2 Model

Following the notation used in Chapter 3, let $\mathcal{S} = \{1, \ldots, \mathrm{S}\}$ be the set of source nodes. The set O_c^s denotes the core overlay nodes used to establish *alternative* paths between a source $s \in \mathcal{S}$ and its destination node(s) in D^s. Let $O_c = \{O_c^s, s \in \mathcal{S}\}$ denote the overall set of core overlay nodes in the network[2].

We consider the overlay selection problem as an offline or a slow time scale (compared to that of the routing problem) online process. Specifically, we assume that the set of source nodes, their corresponding set of destination node(s) as well as the traffic load between each pair are fixed over the timescales that we consider for

[2]This modeling allows us to have different overlay nodes to be assigned for different sources. However, in the following discussion we will assume $O_c^s = O_c^{s'}$ for all $s, s' \in \mathcal{S}$ for simplicity.

the overlay selection problem. Assuming that significant changes to these parameters occur only on a slow time scale, the new set of overlay nodes can be obtained once the set of source and destination nodes and the corresponding traffic matrix change.

Naturally, we use the same objective function used in the previous chapters. We formulate the problem of overlay selection as an optimization problem with the objective function $C(\boldsymbol{x})$, where $C(.)$ is a convex function with respect to $\boldsymbol{x}$, the vector of path rates. However, as the decision variable in the overlay problem is O_c but not $\boldsymbol{x}$ we redefine the objective function as $V(O_c)$, whose value is equal to optimal cost $C(\boldsymbol{x^*})$, *i.e.,* $V(O_c) = C(\boldsymbol{x^*})$. Note that $C(\boldsymbol{x^*})$ is obtained by using the generalized routing algorithm presented in Chapter 3 assuming the alternative paths created by the overlay set O_c remain fixed.

As discussed in the previous chapters, due to the nature of the problem, we assume that the analytical structure of $V(O_c)$ cannot be obtained where one can only rely on measurements of the $V(O_c)$ possibly under noise. For that matter, the problem naturally falls into the class of stochastic discrete optimization problems. Let $\tilde{\mathcal{O}}$ be the discrete set of all possible overlay alternatives. Our objective is to find an overlay configuration O_c from $\tilde{\mathcal{O}}$ that minimizes $V(O_c)$,

$$\min_{O_c \in \tilde{\mathcal{O}}} V(O_c). \tag{4.1}$$

The solution to this problem needs not be unique where a globally optimal overlay set $\tilde{\mathcal{O}}^*$ satisfies

$$\tilde{\mathcal{O}}^* = \left\{ O_c \in \tilde{\mathcal{O}} \mid V(O_c) \leq V(O_c') \, \forall \, O_c' \in \tilde{\mathcal{O}} \right\}. \tag{4.2}$$

The optimization problem stated above can be solved using different algo-

rithms. Simulated Annealing (SA) is one such alternative. As stated in [16], for SA algorithm to perform well, it requires a good neighborhood structure and accurate estimates of the objective function values. Otherwise, a poor choice of a neighborhood structure and the use of rough estimates of the objective function values can lead to poor performance (See [23], [16] and references in there). An alternative to SA is the Stochastic Ruler (SR) algorithm [47]. The SR algorithm compares the objective function estimates with a uniformly distributed random number called stochastic ruler. The range of the random number is selected by the range of the objective function. The algorithm tries to maximize the probability that the estimated objective function is smaller than the ruler by iteratively changing the configuration. It is shown that under fairly general conditions SR algorithm converges in probability to the optimum. However, when the range of the objective function is not known in advance, the performance of the algorithm can degrade as too big a ruler reduces the sensitivity of the algorithm and slows down the optimization process, while a small ruler may not be able to distinguish the best solutions from other good solutions that are outside the range of the ruler [16]. Furthermore, similar to the SA, the SR algorithm also requires a neighborhood structure.

4.3 Stochastic Comparison Algorithm

In [16], Gong, Ho and Zhai propose the Stochastic Comparison (SC) algorithm. SC is a general method to solve discrete stochastic optimization problems with large unstructured search spaces. Unlike SA and SR, SC algorithm does not require

any neighborhood structure. Experimental results presented in [16] show that the algorithm converges to a good solution very quickly, even under very noisy estimates of the objective function. The algorithm is defined in [16] as follows:

Let $\tilde{V}(\cdot)$ be a sample estimate of the objective function $V(\cdot)$ and $R : \tilde{\mathcal{O}} \times \tilde{\mathcal{O}} \rightarrow [0, 1]$ be a generating probability for $\tilde{\mathcal{O}}$ such that

(a) $\sum_{j \in \tilde{\mathcal{O}}} R(i, j) = 1$ for $i \in \tilde{\mathcal{O}}$,

(b) $R(i, j) > 0$ if and only if $j \in \tilde{\mathcal{O}} \setminus \{i\}$.

Furthermore, let X_k be the overlay configuration selected at iteration k, and M_k be the number of sample estimates that must be obtained for a configuration at the k^{th} iteration (a.k.a testing sequence). Finally, let $\tilde{V}_l(i)$, $l = 1, \ldots, M_k$ denote the M_k samples obtained for some configuration $i \in \tilde{\mathcal{O}}$.

Stochastic Comparison Algorithm (SC)

Initial Data: R, M_k, $i_0 \in \tilde{\mathcal{O}}$.

Step 0: Set $X_0 = i_0$ and $k = 0$.

Step 1: Given $X_k = i$, choose a candidate Z_k from $\tilde{\mathcal{O}} \setminus \{i\}$ with probability

$$P[Z_k = j | X_k = i] = R(i, j), \quad j \in \tilde{\mathcal{O}} \setminus \{i\}. \tag{4.3}$$

Step 2: Given $Z_k = j$, set

$$X_{k+1} = \begin{cases} Z_k, & \text{if } \tilde{V}_l(j) < \tilde{V}_l(i) \text{ for all } l = 1, \ldots, M_k, \\ X_k, & \text{otherwise} \end{cases} \tag{4.4}$$

Step 3: Set $k = k + 1$ and go to Step 1.

The SC algorithm is proved to converge to the optimal set $\tilde{\mathcal{O}}^*$ under the following conditions:

(a) The estimates $\tilde{V}(i)$ are unbiased *i.e.,* $E[\tilde{V}(i)] = V(i)$ and identically distributed;

(b) The variance of the estimates is finite *i.e.,* $E[\tilde{V}(i) - E[\tilde{V}(i)]]^2 < \infty$.

(c) $R(i, j) > 0$ for all $j \neq i$, $j \in \tilde{\mathcal{O}}$.

(d) The testing sequence M_k satisfies $M_k = \lfloor c \log_\sigma(k + k_0 + 1) \rfloor$, $k = 0, 1 \ldots$, for some positive numbers c, σ, and k_0.

Returning back to the overlay selection problem, let us define the measurement error in observing the network cost as $\mu(O_c)$ under an overlay configuration O_c. Then,

$$\tilde{V}(O_c) = V(O_c) + \mu(O_c). \tag{4.5}$$

Given the conditions for convergence of SC above, the following assumptions guarantee the convergence of the SC algorithm for the overlay selection problem defined in (4.1)

A9. $\mu(O_c)$ is i.i.d with a symmetric continuous probability density function.

A10. $E[\mu(O_c)] = 0$ and $E[\mu(O_c)^2] < \infty$ for all $O_c \in \tilde{\mathcal{O}}$.

Note that in order for the SC algorithm to converge, the testing number M_k should grow logarithmically. However, the log function increases fast at the begin-

ning of the algorithm which requires one to obtain a significant number of observations of the objective function. Moreover, as the SC algorithm progresses, it can potentially test bad overlay configurations that can lead to very bad system performance. Hence, SC algorithm can lead to severe performance variations during its convergence period. Therefore, it is impractical to consider using SC as an online algorithm. This leads us to look for an alternative, possibly suboptimal, solution that is comparably less demanding and is more suitable as an online algorithm.

4.4 A suboptimal heuristic

As discussed in the previous section, it is of interest to obtain an alternative algorithm that can be used online while performing close enough to the optimal solution. We consider an algorithm that adds in sequence the best overlay node found over the set of all nodes that have overlay capability. For simplicity the algorithm ignores the stochastic nature of the problem and therefore is prone to errors resulting from the estimation of the objective function. Furthermore, the algorithm only considers the case where the overlay set is common for all sources, *i.e.*, $O_c^s = O_c^{s'} = O_c$, for all s, $s' \in \mathcal{S}$. Let N_c be the number of core overlay nodes to be added and $O_c(k)$ be the overlay node to be added to the overlay set O_c. The algorithm can be described as follows:

Greedy Heuristic Algorithm (GH)

Initialization: The overlay set $O_c = \emptyset$, the set of network nodes $\mathcal{N}$.

Step 1: $O_c \leftarrow O_c \cup \{\text{argmin}_{i \in \mathcal{N} \setminus O_c} \tilde{V}(O_c \cup \{i\})\}$.

Step 2: If $|O_c| < N_c$ go to Step 1.

The algorithm defined above is greedy since at each step it tries to add the best overlay node that it can find. It is easy to see that the complexity of the algorithm is $O(|\mathcal{N}|^{N_c})$. Given the fact that one prefers to keep N_c as small as possible for the reason discussed in Section 4.1, this algorithm can be viewed as an online algorithm that operates at a slower time scale compared to that of the routing problems considered in Chapter 2 and Chapter 3. We will refer to this algorithm as Greedy Heuristic (GH) in Section 4.5 where we compare the performance of GH to SC using simulations under different network topologies and traffic types.

A distinct feature of GH is that we can add overlay nodes to the network in a gradual manner if GH is employed for the overlay selection process. This is due to the greedy nature of the algorithm and can be seen from **Step 1** of the algorithm. In other words, once a node is selected, it will not be removed in the following iterations. Therefore, this allows us the opportunity to set N_c initially to a small value, and gradually increase it to eliminate network congestion without the need to replace the overlay nodes that are already active. Note also that the performance of GH algorithm can only get better no matter which node is added to the existing overlay set. This is important for the online implementation of the algorithm.

4.5 Simulation Results

In this section we evaluate the performance of both the SC based optimal algorithm and the proposed heuristic solution GH.

For the optimization problem, we have selected the cost function to be $V(O_c) = C(\boldsymbol{x}^*) = \max_{l \in L} C^l(\boldsymbol{x}^*)$ and the link cost function $C^l(.)$ is defined as $\left(x^l/c^l\right)^2$, where c^l is the link capacity and x^l is the link rate as defined in Section 3.2. This formulation minimizes the maximum link utilization in the network, which is one of the major indicators of network congestion.

In order to get the cost estimates $\tilde{V}(O_c)$ for a particular overlay set, O_c, we make use of our SPSA based algorithms presented in Chapters 2 and 3 and allow the algorithm to converge by running the algorithms over a long period of time, namely 2,500 seconds. We have used a constant step size policy with step size $a_s(k) = a = 0.3$ for all the SPSA based routing algorithms (both for unicast and multicast cases).

For the SC algorithm to converge to the optimal solution the testing sequence M_k, *i.e.,* the number of sample estimates that must be obtained for a configuration at the k^{th} iteration should satisfy $M_k = \lfloor c \log_\sigma(k + k_0 + 1) \rfloor$, $k = 0, 1 \ldots$, for some positive numbers c, σ, and k_0. However, the log function increases fast at the beginning of the algorithm. This requires one to obtain a significant number of observations of the objective function. Therefore, as in [16] we use a linear sequence, $M_k = 1 + \lfloor k/500 \rfloor$, which is a reasonable approximation to the logarithm sequence over the finite range of k used in the experiments. We constrain the optimization algorithm by fixing the number of overlay nodes N_c in order to observe the effect of every additional overlay node in minimizing the network cost. Hence, the SC algorithm returns the best set of overlay nodes with a cardinality of N_c. The generating function $R(i, j)$ is set to be the reciprocal of the total number of overlay set

configurations. This way, given the current set of overlay nodes, the probability of selecting any overlay set except the current set as a candidate is the same. A major source of noise in estimation comes from the random nature of the traffic sources. In this study, we assume that each source node generates traffic according to a Poisson process.

Experiments are conducted with two different network topologies with both unicast and multicast sessions.

4.5.1 Unicast traffic sessions under first network topology

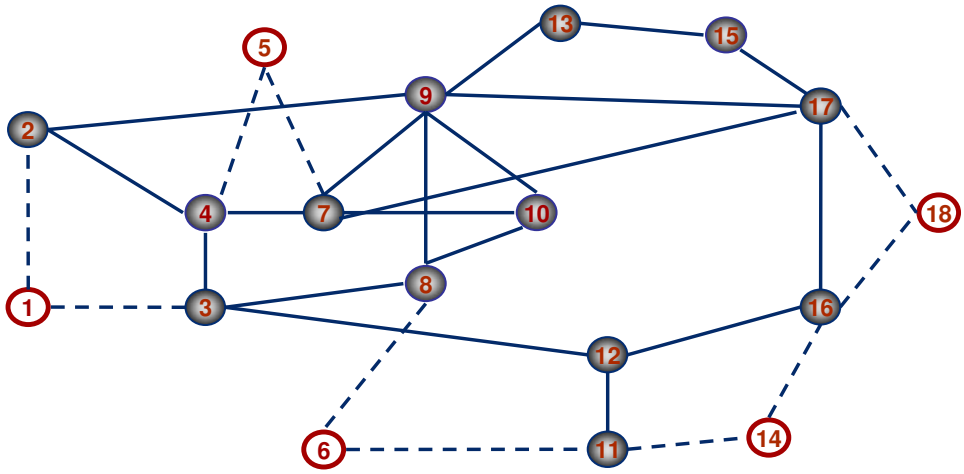


Figure 4.1: Network topology 1

The first topology is shown in Figure 4.1. This topology is also used in [3, 32] and closely resembles the MCI backbone topology reported in [30]. Each link has a capacity of 20 Mbps. Packet size is 500 bytes. Nodes 1, 5, 6, 14 and 18 serve as both source and destination nodes. This gives us a total of 20 SD pairs. Each source node generates Poisson traffic at an average rate of 11.5 Mbps. We demonstrate the

convergence performance of the SC algorithm under different values of N_c in Figure 4.2. We see that while the maximum link utilization in the network decreases as we increase the number of overlay nodes, *i.e.,* the number of paths available to each SD pair, the benefit obtained from a new overlay node gets smaller with each additional overlay node.

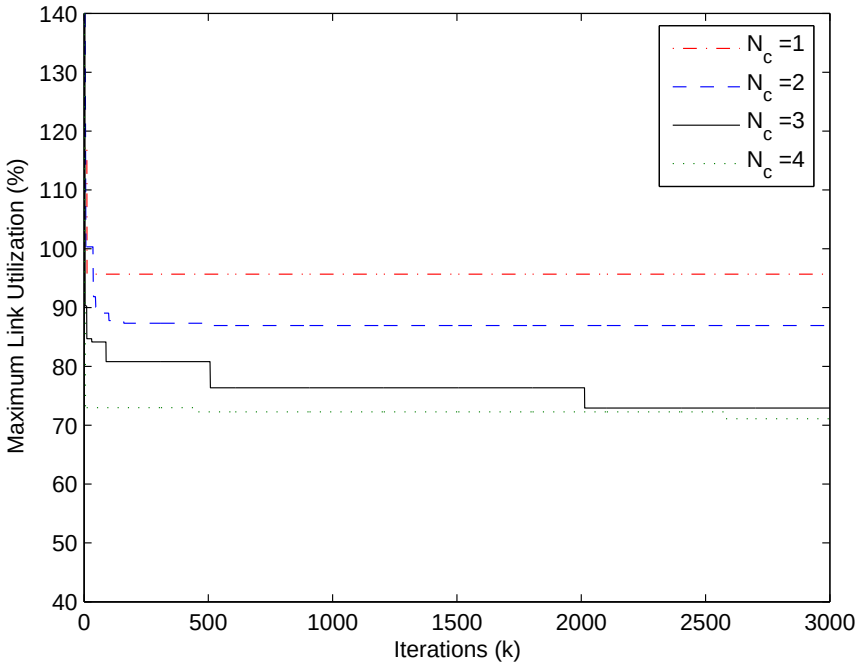


Figure 4.2: Variation of maximum link utilization in the first topology under stochastic comparison algorithm.

In Table 4.1, we present the performance of our heuristic solution GH compared to that of the SC algorithm. We observe that the heuristic solution obtains exactly the same set of overlay nodes as the SC algorithm. Note that cost values $\tilde{V}(O_c)$ are not equivalent due to the stochastic nature of the problem as sample paths are not identical for the two algorithms considered while obtaining the cost estimate $\tilde{V}(O_c)$.

Stochastic Comparison			Greedy Heuristic		
N_c	O_c	$\tilde{V}(O_c)$	N_c	O_c	$\tilde{V}(O_c)$
1	$\{10\}$	0.915	1	$\{10\}$	0.915
2	$\{9, 10\}$	0.756	2	$\{9, 10\}$	0.754
3	$\{9, 10, 14\}$	0.532	3	$\{9, 10, 14\}$	0.531
4	$\{9, 10, 14, 15\}$	0.505	4	$\{9, 10, 14, 15\}$	0.511

Table 4.1: Performance comparison of GH vs. SC.

This result strongly supports the GH against the SC algorithm due to its simplicity. However, one cannot claim the optimality of GH as GH ignores the stochastic nature of the problem, and hence is prone to errors. On the other hand, it might be possible for GH to be optimal for the corresponding deterministic problem where all random variables are replaced with their mean values. This is an open problem for us to consider as a future work.

4.5.2 Unicast traffic sessions under second topology

Figure 4.3 shows the second topology we consider. It is a close approximation of Sprint backbone topology reported in [44]. Compared to the first topology with an average node degree of 3.1667, the second topology is more densely connected and has an average node degree of 5.0769. The capacity of a link is 20 Mbps.

Nodes 4, 6, 8, 10, 15, 18 and 21 serve as both source and destination nodes. This gives us a total of 42 SD pairs. Each source node generates Poisson traffic at an average rate of 8.5 Mbps for each destination. We demonstrate the convergence performance of the SC algorithm under different values of N_c in Figure 4.4. Similar

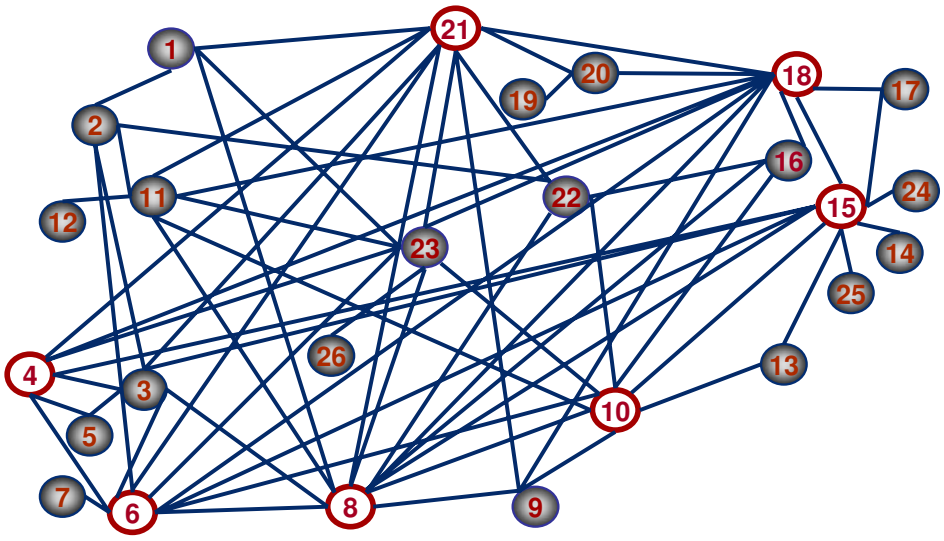


Figure 4.3: Network topology 2

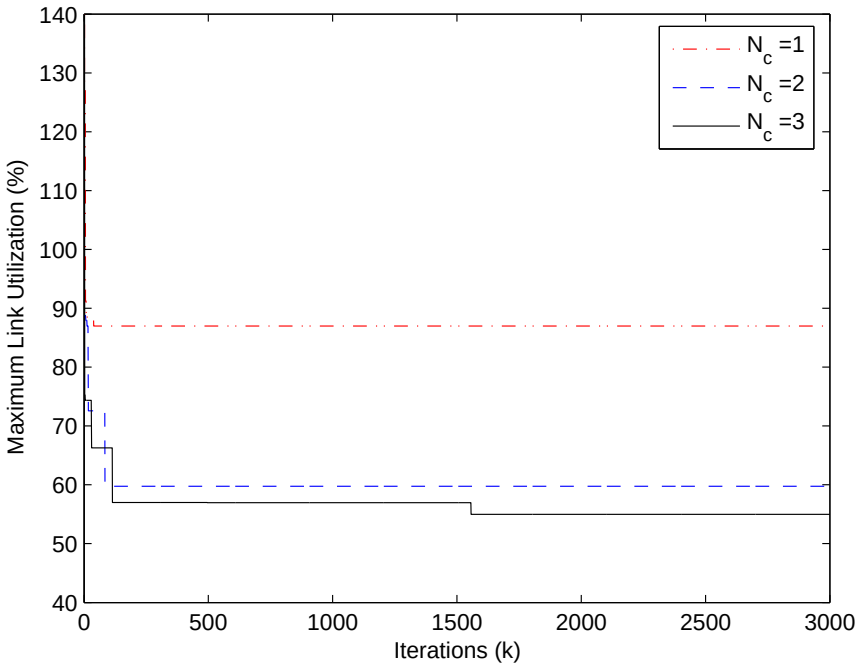


Figure 4.4: Variation of maximum link utilization in the second topology under stochastic comparison algorithm.

Stochastic Comparison			Greedy Heuristic		
N_c	O_c	$\tilde{V}(O_c)$	N_c	O_c	$\tilde{V}(O_c)$
1	$\{11\}$	0.757	1	$\{11\}$	0.757
2	$\{16, 23\}$	0.357	2	$\{11, 23\}$	0.358
3	$\{13, 16, 23\}$	0.302	3	$\{11, 13, 23\}$	0.295

Table 4.2: Performance comparison of GH vs. SC.

to the results obtained in subsection 4.5.1, we observe that a limited number of overlay nodes is sufficient to eliminate congestion in the network and addition of further overlay nodes brings relatively low gains.

We compare the performance of the GH heuristic to SC based algorithm in Table 4.2. Unlike the results obtained for the first topology, the set of core overlay nodes obtained from both algorithms are not identical. However, they still seem to be very close. Recall that the SC algorithm converges asymptotically and it requires a logarithmic testing sequence M_k, which is not utilized in these experiments due to practicality concerns. Hence, one cannot claim the optimality of the results obtained from the SC algorithm. As a consequence of this fact and due to the fact that the GH algorithm appears to manage to eliminate the congestion as well as the SC algorithm, we conclude that the GH can be a good alternative to the SC based solution for locating the core overlay nodes.

4.5.3 Unicast traffic sessions with a non-uniform traffic matrix

In this subsection, we investigate the effect of variation in the input traffic rates on the optimal set of core overlay nodes. We use the second topology and the corresponding source nodes aforementioned. However, instead of having a uniform traffic generation as done in Section 4.5.2, here we assume that each source node generates Poisson traffic with a different average rate. Specifically, we assume that nodes 4, 6, 8, 10, 15, 18 and 21 generate traffic with an average rate of 6.4 Mbps, 8.5 Mbps, 10.2 Mbps, 8.5 Mbps, 3.8 Mbps, 8.9 Mbps and 8.2 Mbps respectively.

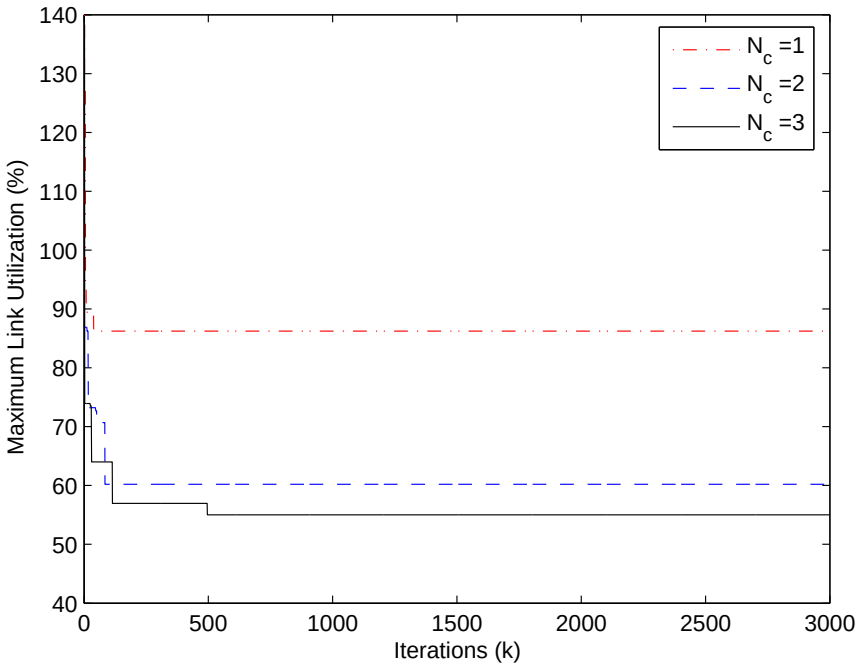


Figure 4.5: Variation of maximum link utilization in the second topology under stochastic comparison algorithm.

Convergence performance of the SC algorithm for different values of N_c is depicted in Figure 4.5. The set of overlay nodes obtained under SC and GH

Stochastic Comparison			Greedy Heuristic		
N_c	O_c	$\tilde{V}(O_c)$	N_c	O_c	$\tilde{V}(O_c)$
1	$\{11\}$	0.744	1	$\{11\}$	0.744
2	$\{16, 23\}$	0.363	2	$\{11, 23\}$	0.360
3	$\{13, 22, 23\}$	0.303	3	$\{11, 13, 23\}$	0.281

Table 4.3: Performance comparison of GH vs. SC.

algorithms are given in Table 4.3. We observe that the set of overlay nodes the GH algorithm obtains is identical to the set of overlay nodes it selects under the uniform traffic matrix presented in Section 4.5.2. This is also true for the SC algorithm except for the case $N_c = 3$. This result suggests that minor changes in the traffic distribution do not have a significant effect on the selection of the overlay nodes. Hence, it is fair to claim that a service provider employing overlays as a means of providing multiple paths does not need to change or relocate the core overlay nodes unless there is a dramatic change in the traffic pattern. This is an important feature for the practical deployment of overlay nodes for the purpose of establishing paths in an intra-domain network.

4.5.4 Multicast traffic sessions

In this subsection we evaluate the performance of proposed algorithm under multicast traffic. Again we employ the second topology given in Figure 4.3. There are three multicast sources and $\mathcal{S} = \{1, 9, 22\}$. Each source has 18 receivers and generates Poisson traffic with an average rate of 8.5 Mbps.. The receiver sets are

given by $D^1 = \{2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 15, 16, 19, 20, 21, 22, 23, 25\}$, $D^9 = \{1, 2, 3, 4, 6, 7, 8, 10, 11, 13, 15, 16, 17, 18, 21, 22, 23, 24\}$ and $D^{22} = \{1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 15, 16, 17, 20, 21, 23, 26\}$. We employ the NM-IIb presented in Chapter 3 as the multipath multicast routing algorithm.

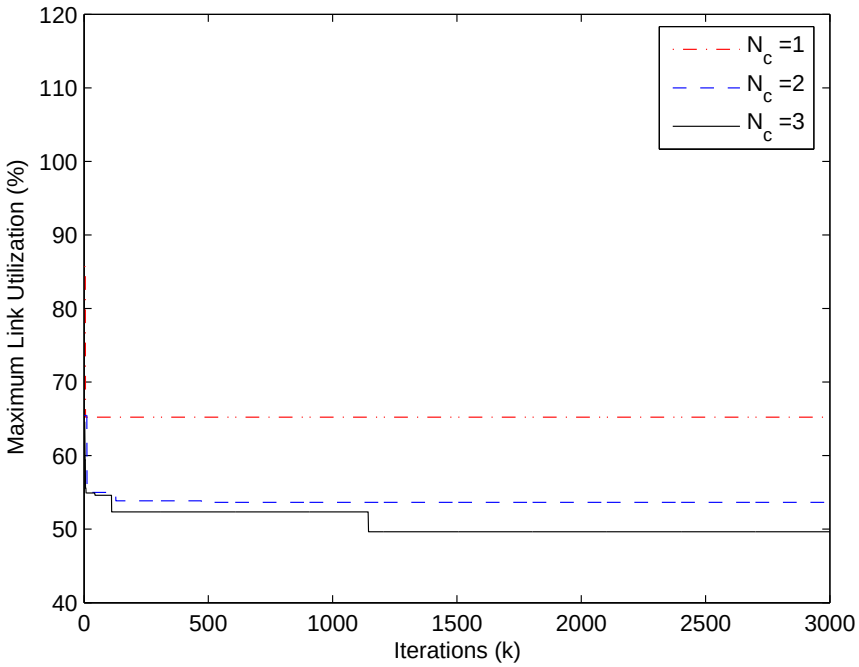


Figure 4.6: Variation of maximum link utilization in the second topology under stochastic comparison algorithm for multicast traffic.

Figure 4.6 depicts the performance of the SC algorithm for different values of N_c. The set of core overlay nodes that SC and GH algorithms return with their corresponding objective function values is given in Table 4.4. Unlike the unicast traffic case, we see that the set of overlay nodes given by each algorithm under multicast traffic is not similar. However, the objective function values for both cases are still very close. This suggests that the optimal overlay node set may not

Stochastic Comparison			Greedy Heuristic		
N_c	O_c	$\tilde{V}(O_c)$	N_c	O_c	$\tilde{V}(O_c)$
1	$\{13\}$	0.425	1	$\{13\}$	0.425
2	$\{5, 10\}$	0.286	2	$\{13, 23\}$	0.290
3	$\{5, 10, 21\}$	0.247	3	$\{6, 13, 23\}$	0.282

Table 4.4: Performance comparison of GH vs. SC. for multicast traffic

be unique, *i.e.,* the cardinality of $\tilde{\mathcal{O}}^*$ may be larger than one. Nevertheless, the simulation results once again show that the GH algorithm can substitute for the SC algorithm in practice without sacrificing network performance under different network settings and under different traffic types.

Chapter 5

Conclusion

In this thesis we have addressed a fundamental traffic engineering problem, namely traffic mapping, *i.e.,* load balancing the network traffic flows along multiple paths.

Motivated by increasing demand for services requiring higher data rates, we considered the network congestion and resource utilization problem in a dynamical network environment and formulated the problem in an optimization framework. In order to reflect practical constraints of a real network situation, we have considered optimal multi-path routing in environments where the link cost derivatives can be estimated, but for which an analytic expression may not exist. The main contributions in this thesis are as follows.

- **Measurement based optimal multipath routing for unicast sessions**

  We focused our analysis on intra-domain networks with unicast traffic sources. We have established a distributed measurement based optimal multipath routing algorithm which is based on simultaneous perturbations. We have mathematically proven the optimality and stability of our routing scheme. We analyzed the convergence properties of the proposed algorithm both with decreasing step sizes and with a fixed step size. We demonstrated that an SPSA algorithm provides significant improvements over an algorithm based on tradi-

tional finite-difference methods. Specifically, we have shown that our scheme results in much shorter measurement periods during the gradient estimation phase, and as a result, converges faster. Furthermore, using simulations we have shown that our scheme can quickly alleviate network congestion and distribute load efficiently under dynamic network conditions.

- **Measurement based optimal multipath routing for multicast sessions**

  We next generalized our study in an effort to establish a unifying framework for multipath routing of both unicast and multicast traffic within a domain. We developed a distributed optimal routing algorithm that balances the load along multiple paths for multiple multicast and/or unicast sessions. As in the unicast only case, our *measurement-based* algorithm does not assume the existence of the gradient of an analytical cost function and is a natural generalization of the *unicast* routing algorithm. To the best of our knowledge this is the first attempt to address the issue of (optimal) multipath routing with *multiple* multicast sessions in a *distributed* manner, while relying only on (local) network measurements. In order to evaluate the performance of our routing scheme, we considered three different network models (NM-I, II, and III) to quantify the benefits of additional functionality/intelligence in the underlying IP network. In NM-III we established a routing framework that generalizes the multiple distribution trees to a more general multiple path scenario where each destination can receive packets at a different rate from a multicast tree. The need for complicated bookkeeping at the sources and intermediate IP

routers is handled by employing Digital Fountain coding. As for the unicast case, we proved the convergence properties of the multicast routing algorithm both with decreasing step sizes and with a fixed step size. Further, our simulation studies have shown that while basic IP multicast functionality in NM-II is crucial for better performance, additional functionalities introduced in NM-III provide only marginal benefits in relation to required complexity.

- **Path optimization through overlay node placement**

  The performance of the routing algorithms are limited by the paths that are available. We considered an overlay architecture for providing multiple paths in an IP network. We have addressed this problem by formulating a discrete stochastic optimization problem where we attempt to place a limited number of overlay nodes in the network to establish alternative paths. We have solved this optimization problem using the Stochastic Comparison (SC) [16] algorithm. However, SC is computationally demanding. Moreover, in order to converge to the optimal set of overlay nodes, it needs to try and evaluate many other set of overlay nodes configurations, some of which may be far worse than another and may disrupt network performance if SC is used as an online setting. Motivated by those facts, we have also considered suboptimal solutions suitable for dynamical network environments and presented a *greedy* heuristic solution. We have also shown through simulations that the performance of the heuristic algorithm is comparable to that of SC algorithm.

BIBLIOGRAPHY

[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, Network Information Flow, *IEEE Transactions on Information Theory*, **IT-46** (2000) 1204–1216.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek and R. Morris, Resilient overlay networks,*Proc. 18th ACM Symp. on Operating Systems Principles (SOSP)* (2001).

[3] G. Apostolopoulos, R. Guerin, S. Kamat and S. Tripathi, Quality of service based routing: a performance perspective, *ACM SIGCOMM* (1998).

[4] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, Overview and principles of internet traffic engineering, *RFC 3272* (2002).

[5] D. Bertsekas and R. Gallager, Data networks, *Prentice-Hall Inc., 2nd edition* (1992).

[6] D. Bertsekas, A. Nedic and A.E. Ozdaglar, Convex Analysis and Optimization, *Athena Scientific* (2003).

[7] J. R. Blum, Multidimensional stochastic approximation methods, *Ann. Math. Stat.* **25** (1954) 737–744.

[8] K. Claffy, G. Miller and K. Thompson, The nature of the beast: recent traffic measurements from an internet backbone, *Internet Society's Networking Conference (INET)* (1998).

[9] A. Collins, The detour framework for packet rerouting, *PhD thesis,* (1998).

[10] A. Elwalid, C. Jin, S. Low and I. Widjaja, MATE: Multipath adaptive traffic engineering, *Computer Networks-The International Journal of Computer and Telecommunications Networking* **40** (2002) 695–709.

[11] A. Elwalid, C. Jin, S. Low and I. Widjaja, MATE: MPLS adaptive traffic engineering, *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (2001).

[12] G. B. Folland, Real Analysis: Modern Techniques and Their Applications, *John Wiley & Sons*, (1984).

[13] B. Fortz and M. Thorup, Internet traffic engineering by optimizing OSPF weights, *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (2000).

[14] M. Fu and S. D. Hill, Optimization of discrete event systems via simultaneous perturbation stochastic approximation, *Transactions of the Institute of Industrial Engineers* **29** (1997) 223–243.

[15] E. Gelenbe, On approximate computer system models, *Journal ACM* **22** (1975) 261–269.

[16] W.-B. Gong, Y.-C. Ho and W. Zhai, Stochastic Comparison Algorithm for Discrete Optimization with Estimation, *SIAM Journal of Optimization* **10** (1999) 384–404.

[17] G.R. Grimmett and D.R. Stirzaker, Probability and random processes, *Oxford Science Publications, 2nd edition* (1998).

[18] T. Güven, R. J. La, M. A. Shayman and B. Bhattacharjee, Measurement based optimal routing on overlay architectures for unicast sessions, *Special Issue of Computer Networks Journal on Network Modeling and Simulation*, to appear.

[19] T. Güven, C. Kommareddy, R. J. La, M. A. Shayman and B. Bhattacharjee, Measurement Based Optimal Multi-path Routing, *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (2004).

[20] Y. He, M. C. Fu and S. I. Marcus, Convergence of Simultaneous Perturbation Stochastic Approximation for Nondifferentiable Optimization, *IEEE Transactions on Automatic Control*, **48** (2003) 1459–1463.

[21] D. P. Heyman and T. V. Lakshman, What are the implications of long range dependence for VBR video traffic engineering, *IEEE/ACM Transactions on Networking* **4(3)** (1996) 301-317.

[22] D. P. Heyman and M. J. Sobel, Stochastic models in operations research, *McGraw-Hill* (1982).

[23] R. Jayaraman and F. Darema, Error tolerance in parallel simulated annealing techniques, *Proceedings of International Conference on Computer Design, IEEE Computer Society Press*, (1988).

[24] J. Kiefer and J. Wolfowitz, Stochastic estimation of a regression function, *Ann. Math. Stat.* **23** (1952) 462–466.

[25] R. Koetter and M. Medard, Beyond Routing: An Algebraic Approach to Network Coding, *IEEE Infocom* (2002).

[26] C. Kommareddy, T. Güven, B. Bhattacharjee, R. J. La and M. A. Shayman, Overlay routing for path multiplicity, *Tech. Rep. UMIACS-TR# 2003-70* <http://www.cs.umd.edu/Library/TRs/CS-TR-4500/CS-TR-4501.pdf>.

[27] H.J. Kushner and G.G. Yin, Stochastic approximation algorithms and applications, *Springer-Verlag* (1997).

[28] H.J. Kushner and D.S. Clark, Stochastic approximation methods for constrained and unconstrained systems, *Springer-Verlag*, (1978).

[29] S-Y. R. Li and R. W. Yeung, Linear Network Coding, *IEEE Transactions on Information Theory* **49** (2003) 371–381.

[30] Q. Ma and P. Steenkiste, On path selection for traffic with bandwidth guarantees, *IEEE International Conference on Network Protocols* (1997).

[31] D. J. C. Mackay, Information Theory, Inference, and Learning Algorithms, *Cambridge University Press* (2003).

[32] S. Nelakuditi and Z. L. Zhang, A localized adaptive proportioning approach to QoS routing, *IEEE Commun. Mag.* **40** (2002) 66–71.

[33] T. Noguchi, T. Matsuda and M. Yamamoto, Performance Evaluation of New Multicast Architecture with Network Coding, *IEICE Trans. Comm.* E86-B (2003) 1788–1795.

[34] K. Park and Y. Shin, Uncapacitated point-to-multipoint network flow problem and its application to multicasting in telecommunication networks, *European Journal of Operational Research* **147** (2003) 405–417.

[35] R. T. Rockafellar,Convex Analysis, *Princeton, NJ: Princeton Univ. Press*, (1970).

[36] M. A. Rodrigues and K. G. Ramakrishnan, Optimal routing in shortest-path networks, *International Telecommunications Symposium (IEEE ITS)* (1994).

[37] P. Sadegh, Constraint optimization via stochastic approximation with a simultaneous perturbation gradient approximation, *Automatica* **33** (1997) 889–892.

[38] K. Sinha and S. Patek, OpIATE: optimization integrated adaptive traffic engineering, *Tech. Rep.* <http://www.sys.virginia.edu/techreps/2002/sie-020001.pdf> (2001).

[39] A. Shokrollahi,Raptor Codes, *preprint*, <http://www.inference.phy.cam.ac.uk/mackay/DFountain.html>, (2003).

[40] J.M. Smith and F.R.B. Cruz, The buffer allocation problem for general finite buffer queueing networks, <http://www.ecs.umass.edu/mie/faculty/smith/> *Unpublished*.

[41] J. C. Spall, Stochastic optimization, stochastic approximation and simulated annealing, *Encyclopedia of Electrical and Electronics Engineering (J.G. Webster, ed.), Wiley, New York* **20** (1999) 529–542.

[42] J. C. Spall, Stochastic optimization and the simultaneous perturbation method, *Proceedings of the Winter Simulation Conference* (1999).

[43] J. C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Trans. Automat. Contr.* **37** (1992) 332–347.

[44] N. Spring, R. Mahajan and D. Wetherall, Measuring ISP Topologies with Rocketfuel, *ACM SIGCOMM*, (2002).

[45] J. N. Tsitsiklis and D. P. Bertsekas, Distributed asynchronous optimal routing in data networks, *IEEE Trans. Automat. Contr.* **31** (1986) 325–332.

[46] D. Waitzman and C. Partridge and S. Steering, Distance Vector Multicast Routing Protocol, *RFC 1075* 1998.

[47] D. Yan and H. Mukai, Stochastic discrete optimization, *SIAM Journal of Control Optim.* **30** (1992) 594–612.

[48] Y. Zhu, B. Li and J. Guo, Multicast with Network Coding in Application-Layer Overlay Networks, *IEEE Journal on Selected Areas in Communications*, **22** (2004) 107–120.